НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ ЗАХИСТУ ІНФОРМАЦІЇ

«На правах рукопису»
УДК 003.26:519.21

«До захисту допущено»
В.о. завідувача кафедрою
_____ М.М.Савчук
(підпис) (ініціали, прізвище)
“___” _____ 2020р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 113 «Прикладна математика»
(код і назва)

на тему: Метод оцінювання стійкості блокових шифрів до криптоаналізу на основі усічених диференціалів

Виконав (-ла): студент (-ка) 6 курсу, групи ФІ-83мн
(шифр групи)

Якимчук Олексій Петрович
(прізвище, ім’я, по батькові) (підпис)

Керівник доцент, кандидат фізико-математичних наук
Яковлєв С.В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

2Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»
Фізико-технічний інститут
Кафедра математичних методів захисту інформації

Рівень вищої освіти: другий (магістерський) за освітньо-науковою програмою

Спеціальність: 113 «Прикладна математика»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедрою
_____ М.М.Савчук
(підпис) (ініціали, прізвище)
«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Якимчук Олексій Петрович
(прізвище, ім'я, по батькові)

1. Тема дисертації Метод оцінювання стійкості блокових шифрів до криптоаналізу на основі усічених диференціалів ,
науковий керівник дисертації Яковлєв Сергій Володимирович, доцент, кандидат фізико-математичних наук ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від _____ р. № _____

2. Термін подання студентом дисертації _____

3. Об'єкт дослідження: інформаційні процеси в системах криптографічного захисту.
4. Предмет дослідження: моделі та методи криптоаналізу блокових шифрів на основі усічених диференціалів.
5. Перелік завдань, які потрібно розробити:
   1) проаналізувати опубліковані результати використання усічених диференціалів для побудови атак на блокові шифри;
   2) запропонувати формалізований підхід до побудови усічених диференціалів;
   3) запропонувати параметр стійкості, який буде характеризувати стійкість шифруючих перетворень до атак на основі усічених диференціалів;
   4) перевірити адекватність запропонованих методів оцінювання на модельних шифрах.

6. Орієнтовний перелік ілюстративного матеріалу: робота містить 2 рисунки та 11 таблиць.

7. Орієнтовний перелік публікацій: результати роботи частково представлено у матеріалах двох міжнародних науково-практичних конференціях.

8. Консультанти розділів дисертації*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Визначення орієнтованої теми дисертації та можливих напрямків дослідження по темі	вересень – жовтень 2018 року	виконано
2.	Опрацювання основної літератури на тему дослідження	листопад 2018 – серпень 2019	виконано
3.	Огляд наявних прикладів застосування атак на основі усічених диференціалів	жовтень – грудень 2019 року	виконано
4.	Дослідження формалізації підходу до побудови усічених диференціалів	січень – лютий 2020 року	виконано
5.	Визначення та перевірка параметрів, які можуть використовуватись для оцінки імовірності усічених диференціалів	лютий – березень 2020 року	виконано
6.	Побудова і програмна реалізація алгоритму пошуку високоімовірнісних усічених диференціалів	квітень 2020	виконано

Студент _____ Якимчук О. П.
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Яковлєв С. В.
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено наукового керівника магістерської дисертації.

РЕФЕРАТ

Кваліфікаційна робота містить: 69 стор., 2 рисунки, 11 таблиць, 13 джерел.

Одним з підходів для застосування диференціального криптоаналізу до блокових шифрів є аналіз усічених диференціалів. Якщо класичний диференціальний криптоаналіз досліджує повну різницю між двома текстами, то диференціальний аналіз, що використовує усічені диференціали, враховує відмінності між текстами, які визначаються лише частково. Такий підхід дозволяє успішно застосовувати диференціальний криптоаналіз до сучасних шифрів, де стандартний підхід не давав жодних результатів. Не зважаючи на те, що диференціальний криптоаналіз з використанням усічених диференціалів вперше запропонований ще у 1995, досі не існує формальної теорії, яка його описує та дозволяє проводити оцінку стійкості конкретних шифрів до нього.

Метою цієї роботи є розробка формалізованого підходу до використання усічених диференціалів. Об'єктом дослідження є інформаційні процеси в системах криптографічного захисту. Предмет дослідження — моделі та методи диференціального криптоаналізу блокових шифрів.

В даному дослідженні запропоновано два формалізованих підходи до побудови усічених диференціалів. Для кожного з них запропоновано параметр стійкості, який характеризує імовірність усіченого диференціала. Для кожного з параметрів наведено характеристики та властивості. Також в роботі запропоновано алгоритм пошуку високоімовірнісних усічених диференціалів, який був успішно застосований на модельному шифрі.

ДИФЕРЕНЦІАЛЬНИЙ КРИПТОАНАЛІЗ, УСІЧЕНІ ДИФЕРЕНЦІАЛИ, ІМОВІРНІСТЬ ДИФЕРЕНЦІАЛУ, БЛОКОВІ ШИФРИ

ABSTRACT

The qualifying paper contains: 69 pages, 2 figures, 11 tables, 13 sources.

One of the approaches for applying differential cryptanalysis to block ciphers is the analysis of truncated differentials. If classical differential cryptanalysis investigates the complete difference between two texts, then truncated differential analysis investigates differences between the texts, which are only partially determined. This approach allows to use differential cryptanalysis to modern ciphers, to which classical differential cryptanalysis is not applicable. Despite truncated differential cryptanalysis was suggested in 1995, for today there is no formalized theory that describes truncated differential cryptanalysis and allows evaluate ciphers security against it.

The purpose of this work is to develop formalized approach for truncated differential cryptanalysis. The object of research is the information processes in cryptographic security systems. Subject of research — models and methods of differential cryptanalysis of block ciphers.

In this work were presented two formalized approaches for creating truncated differentials. For each of them suggested security parameter that shows truncated differential probability. Also in this work was presented algorithm for search truncated differentials with high probability. The algorithm was successfully applied to model cipher.

DIFFERENTIAL CRYPTANALYSIS, TRUNCATED DIFFERENTIALS, DIFFERENTIAL PROBABILITY, BLOCK CIPHERS

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

$\oplus$ — операція побітового додавання

V_n — простір бітових векторів довжини n

V_n^* — простір векторів довжини n, елементами яких можуть бути $\{0, 1, *\}$

$[P]$ — дужки Айверсона: $[P]$ дорівнює 1, якщо P – істинне, та дорівнює 0, якщо P – хибне

$DP^f(\alpha, \beta)$ — імовірність диференціалу (α, β) функції f

$D^f(\alpha, \beta)$ — множина вхідних текстів, які з вхідною різницею α дають вихідну різницю β для функції f

$\Delta(\alpha)$ — множина усіх можливих різниць для певної маски α

$TDP^f(\alpha, \beta)$ — імовірність усіченого диференціалу (α, β) функції f

$TD^f(\alpha, \beta)$ — множина двійкових векторів, які з вхідної різниці маски α, дають вихідну різницю маски β

$<<<$ — циклічний зсув вліво на деяку кількість бітів

ВСТУП

Диференціальний криптоаналіз став відомий широкому загалу на початку 1990-х років, після публікацій Біхама та Шаміра [1]. Цей метод виявився ефективним проти деяких відомих на той час симетричних шифрів. В 1995 році Ларс Кнудсен запропонував дещо змінений підхід до диференціального криптоаналізу, використовуючи усічені диференціали. Це допомогло застосувати диференціальний криптоаналіз до шифрів, де стандартний підхід не давав жодних результатів. Якщо звичайний диференціальний криптоаналіз досліджує повну різницю між двома текстами, то диференціальний аналіз, що використовує усічені диференціали, враховує відмінності між текстами, які визначаються лише частково.

Не зважаючи на відносно успішні випадки практичного застосування даного методу, на сьогодні не існує формальної теорії, яка описує криптоаналіз на основі усічених диференціалів та дозволяє проводити оцінку стійкості конкретних шифрів до нього.

Метою роботи розробка формалізованого підходу до використання усічених диференціалів.

Для досягнення мети необхідно виконати такі **завдання**:

1) проаналізувати опубліковані результати використання усічених диференціалів для побудови атак на блокові шифри;

2) запропонувати формалізований підхід до побудови усічених диференціалів;

3) запропонувати параметр стійкості, який буде характеризувати стійкість шифруючих перетворень до атак на основі усічених диференціалів;

4) перевірити адекватність запропонованих методів оцінювання на модельних шифрах.

Об'єкт дослідження: інформаційні процеси в системах

криптографічного захисту.

Предмет дослідження: моделі та методи криптоаналізу блокових шифрів на основі усічених диференціалів.

Методи дослідження, які використовувались при розв'язанні поставлених завдань: методи теорії імовірності, лінійної та абстрактної алгебри, математичної статистики, комбінаторного аналізу, методи комп'ютерного та статистичного моделювання.

Наукова новизна: у роботі запропоновано формалізований підхід до побудови усічених диференціалів та визначено параметр стійкості, який характеризує імовірність усічених диференціалів.

Практична значущість: результати цієї роботи можуть застосовуватись для оцінювання стійкості ітеративних блокових шифрів до криптоаналізу на основі усічених диференціалів.

Апробація. Результати даної роботи було частково представлено на таких конференціях:

1) XVIII Всеукраїнська наукова конференція студентів, аспірантів та молодих вчених «Теоретичні і прикладні проблеми фізики, математики та інформатики», м. Київ, 12 — 13 травня 2020 року, доповідь «Параметри, які характеризують стійкість S-блоків до аналізу усічених диференціалів»;

2) XII Міжнародна науково-практична конференція «Інтернет — Освіта — Наука 2020», м. Вінниця, 26 — 29 травня 2020 року, доповідь «Параметр, який характеризує стійкість S-блоків до аналізу усічених диференціалів».

1 ДИФЕРЕНЦІАЛЬНИЙ КРИПТОАНАЛІЗ НА ОСНОВІ УСІЧЕНИХ ДИФЕРЕНЦІАЛІВ

В даному розділі розглядаються необхідні теоретичні відомості про диференціальний криптоаналіз в цілому, диференціальний криптоаналіз на основі усічених диференціалів та аналізуються відомі приклади успішного застосування диференціального криптоаналізу на основі усічених диференціалів до сучасних шифрів.

1.1 Основні поняття диференціального криптоаналізу

Позначимо M — множину відкритих текстів, C — множину шифротекстів, K — множину ключів.

Означення 1.1. *Шифруюче перетворення* — це функція вигляду $f : M \times K \to C$, що задовільняє такій умові: для кожного фіксованого значення $k \in K$ перетворення $f(x, k)$ є бієктивним.

Для того, щоб підкреслити, що в шифруючому перетворенні ключ виступає в ролі параметра, використовують позначення $f_k(x)$.

Означення 1.2. *Ітеративний r-раундовий шифр* E — це перетворення виду $E : V_q \times K^r \to V_q$, що є композицією r шифруючих перетворень

$$E = f_{k_r}^{(r)} \left(f_{k_{r-1}}^{(r-1)} \left(\dots \left(f_{k_1}^{(1)}(x) \right) \dots \right) \right).$$

Зауважимо, що в позначенні $f_{k_i}^{i}(x)$, параметр i означає, що це перетворення i-того раунду шифрування. Також тут і надалі будемо вважати, що раундові ключі $k = (k_1, k_2, ..., k_r)$ є випадковими, незалежними та рівномірно розподіленими на ключовому просторі.

Диференціальний криптоаналіз відноситься до *атак останнього раунду*, оскільки ціллю проведення аналізу є відновлення раундового

ключа останнього раунду шифрування — k_r.

Нехай $\times, \bullet$ — деякі операції на просторі V_q. Шифр E можна представити як композицію перетворень, тобто

$$E = F_{1,r-1} \circ f_r,$$

де f_r — шифруюче перетворення останнього раунду, $F_{1,\,r-1}$ — перетворення всіх інших раундів, з першого по $(r-1)$-й. Розглянемо пару відкритих текстів (X, X') таких, що

$$X' = X \times \alpha$$

для деякого фіксованого α, та пару напівшифротекстів (Y, Y') таких, що

$$Y = F_{1,\,r-1}(X) \;\text{ та }\; Y' = F_{1,\,r-1}(X').$$

Нехай для заданого α із високою імовірністю p (більшою ніж імовірність при рівномірному розподілі, тобто $p > 2^{-q}$) виконується рівність $Y' = Y \bullet \beta$ для деякого β, тоді криптоаналітик, якому це відомо, може побудувати статистичний розпізнавач для ключа k_r таким способом:

1. Криптоаналітик накопичує деяку кількість пар випадкових відкритих текстів (X, X') таких, що $X' = X \times \alpha$ та відповідних їм шифротекстів (C, C').

2. Для кожного кандидата в ключі k_r аналітик розшифровує пари (C, C') на один раунд назад та одержує пари (Y, Y').

3. Аналітик перевіряє гіпотезу $Y' = Y \bullet \beta$. Якщо імовірність цієї події близька до p, тоді ключ відгадано правильно; якщо імовірність близька до 2^{-q}, то ключ відгадано неправильно.

Нехай $V_n = \{0,1\}^n$ — простір n-бітових векторів, а $\circ, \bullet$ – операції (не обов'язково різні), кожна з яких визначає структуру абелевої групи з нейтральним елементом на V_n та для них існує нейтральний елемент.

Означення 1.3. *Диференціал функції* f — пара довільних двійкових векторів (α, β). Для диференціала (α, β) позначимо подію

$f(z \circ \alpha) = f(z) \bullet \beta$, що індукується випадковою величиною z, символом $\alpha \underset{f}{\to} \beta$ (або просто $\alpha \to \beta$, якщо функція зрозуміла з контексту).

Для кожного диференціала певної функції є параметр який характеризує важливість цього диференціала, це — імовірність диференціала.

Означення 1.4. *Імовірність диференціала* (α, β) *функції* f — величина

$$DP^f_{\circ,\bullet}(\alpha, \beta) = Pr_z\{\alpha \to \beta\} = \overline{\sum_z}[f(z \circ \alpha) = f(z) \bullet \beta],$$

де $[P]$ — дужки Айверсона: $[P]$ дорівнює одиниці, якщо твердження P істинне, інакше $[P]$ дорівнює нулю.

Для імовірності диференціала можна використовувати позначення $DP^f(\alpha, \beta)$. В такому випадку операції, що використовуються, повинні бути зрозумілі з контексту.

Введемо деяку загальну класифікацію диференціалів:

– *тривіальний диференціал* — диференціал $\alpha = \beta = 0$, всі інші диференціали — *нетривіальні*;

– *неможливий диференціал* — диференціал, імовірність якого дорівнює нулю.

Наведемо властивості диференціальних імовірностей, які використовуються в диференціальному криптоаналізі.

Для булевої функції $f : V_q \to V_q$ виконуються такі співвідношення:

$$DP^f(0, \beta) = [\beta = 0],$$

$$DP^f(\alpha, 0) = [\alpha = 0], \tag{1.1}$$

$$\forall \alpha : \sum_{\beta \in V_q} DP^f(\alpha, \beta) = 1,$$

$$\forall \beta : \sum_{\alpha \in V_q} DP^f(\alpha, \beta) = 1. \tag{1.2}$$

Властивості під номером (1.1) та (1.2) виконуються, коли функція f є бієктивною. У багатьох випадках зручно розглядати множину

$$D^S(\alpha, \beta) = \{x \in V_n : S(x \oplus \alpha) = S(x) \oplus \beta\}.$$

Тоді можна сказати, що $DP^S(\alpha, \beta) = \dfrac{|D^S(\alpha, \beta)|}{2^n}$.

1.2 Диференціальний аналіз на основі усічених диференціалів

Вперше диференціальний аналіз на основі усічених диференціалів був запропонований Ларсом Кнудсеном [2] в 1994 році. Якщо звичайний диференціальний криптоаналіз аналізує повну різницю між двома текстами, то диференціальний аналіз, що використовує усічені диференціали, враховує відмінності між текстами, які визначаються лише частково. Отже, атака робить передбачення лише деяких бітів.

Як відомо, (α, β) називається диференціалом i-того раунду шифрування, якщо різниця α у двох відкритих текстах породжує різницю β у двох шифротекстах після i раундів шифрування. Проте, як зараз буде показано, не завжди необхідно передбачувати шукане значення повністю, інколи достатньо одного біту. Диференціал, за допомогою якого передбачається лише частина шуканого значення називається усіченим диференціалом.

Означення 1.5. Нехай (α, β) диференціал i-того раунду. Якщо α' є підпослідовністю α, а β' є підпослідовністю β, то (α', β') називається *усіченим диференціалом* i-того раунду шифрування. [2]

В даній роботі [2] Кнудсен представив теорему для оцінки складності виконання диференціальної атаки на основі усічених диференціалів. В процесі доведення запропонована загальна схема проведення диференціальних атак з використанням усічених

диференціалів. Така атака може бути розширена для роботи на шифрах з будь-якою кількістю раундів, розраховуючи на всі, крім перших трьох раундових ключів.

Теорема 1.1. *Нехай* $f(x,k) : GF(2^n) \times GF(2^n) \to GF(2^n)$ — *раундова функція в 5-раундовій схемі Фейстеля з розміром блоку* $2n$ *бітів, яка використовує 5 раудових ключів розміром по 5 бітів кожен. Нехай параметр* α $(\alpha \neq 0)$ — *різниця між вхідними відкритими текстами, для якої можливими вихідними різницями є лише частка* W $(0 \leq W \leq 1)$ *від усіх можливих вихідних різниць, тобто*

$$W = \frac{\textit{кількість можливих вихідних різниць для } \alpha}{\textit{кількість усіх можливих вихідних різниць}}.$$

В такому випадку складність диференціальної атаки з використанням усічених диференціалів становить $2L$ *вибраних відкритих текстів та час її виконання близько* $L \times 2^{2n}$, *де* L *найменше ціле число, для якого виконується:* $W^L < 2^{-2n}$. *Значення* L *не перевищує* $2n + 1$.

Доведення. Розглянемо наступну атаку:

1) Нехай α — нетривіальна різниця двох входів для функції f, для якої лише частка W від усіх вихідних різниць є можливою.

2) Обчислити таблицю T (заповнену нулями при ініціалізації), для $i = 0, \ldots, 2^n - 1$:

$$T[f(i) \oplus f(i \oplus \alpha)] = 1.$$

3) Випадково обрати відкритий текст P_1 та обчислити

$$P_2 = P_1 \oplus (\alpha \,||\, 0).$$

4) Отримати шифротексти C_1 та C_2, які відповідають відкритим текстам P_1 та P_2.

5) Для кожного можливого значення k_5 — кандидата на раундовий ключ RK_5 виконати:

а) Розшифрувати C_1, C_2 та один раунд використовуючи

раундовий ключ k_5. Позначити ці напівшифротексти D_1, D_2.

б) Для кожного значення k_4 — кандидата на раундовий ключ RK_4 виконати:

i. Обчислити $t_i = f(D_i^R \oplus k_4)$ для $i = 1, 2$.

ii. Якщо $T[t_1 \oplus t_2 \oplus D_1^L \oplus D_2^L] > 0$, повернути значення k_4 та k_5.

Оскільки нелінійний порядок функції $f(x)$ не може бути більший від $n - 1$, інформацію про вихідні різниці, які отримуються з заданої вхідної різниці, не обов'язково легко визначити. Тому потрібно обрахувати таблицю T для заданої вхідної різниці α; якщо $T[\beta] > 0$, тоді вихідна різниця β є можливою.

Вхідні відкриті тексти для першого раунду є рівними, проте вхідні тексти для другого раунду мають різницю α. Значить, можна обрахувати частку W від усіх можливих різниць текстів четвертого раунду, використовуючи праві частини шифротекстів та значення в таблиці T.

Після закінчення процедури отримано близько $W \times 2^{2n}$ варіантів можливих значень для пари $(RK_4,\ \ RK_5)$, одна з яких є правильною парою ключів. Після достатньої кількості повторювань атаки лише одна пара ключів (правильна пара) буде залишатись завжди запропонованою. Будь-яка інша пара ключів буде попадати в можливі значення ключів з імовірністю W для кожного виконання атаки. Тому, після проведення атаки для L пар відкритих текстів, будь-яка пара ключів, окрім правильної, буде попадати в запропоновані пари завжди з імовірністю W^L. Якщо $W^L < 2^{-2n}$, то з великою імовірністю правильну пару ключів буде знайдено. Оскільки $W \leq 1/2$, то маємо:

$$\min_L : \ \left(\frac{1}{2}\right)^L < 2^{-2n} = 2n + 1,$$

що і потрібно було довести. □

1.3 Приклади успішних атак на основі усічених диференціалів

Оскільки загальну схему диференціальної атаки з використанням усічених диференціалів було введено в доведенні теореми 1.1 вище, тепер можна розглянути декілька прикладів успішного застосування диференціальних атак на основі усічених диференціалів до сучасних шифрів.

Приклад 1.1. *Атака на* 6 *раундів шифру* `DES`. В роботі Кнудсена [2] було сказано, що для шифру `DES` [3] існують усічені диференціали з імовірністю 1. Коли два входи функції F є рівними на вході до S-блоку, то виходи з цього S-блоку є завжди рівними, незалежно від значення входів інших S-блоків. Вихід S-блоку впливає на входи не більше ніж 6ти S-блоків наступного раунду, через структуру перемішувань, які блоки на які не впливають зазначено в таблиці 1.1.

Таблиця 1.1 – Вплив виходів S-блоків на S-блоки наступного раунду шифру `DES`

Вихід S-блоку	S-блоки, на які він не впливає
1	1, 7
2	2, 6
3	3, 1
4	4, 2
5	5, 8
6	6, 4
7	7, 5
8	8, 3

Цей факт можна використати для побудови 4-ох раундового усіченого диференціала з імовірністю 1 для DES, який дає інформацію про різницю 8-ми бітів в шифротекстах після 4-ох раундів шифрування.

Розглянемо пару відкритих тестів, в яких праві частини співпадають, а ліві відрізняються так, що входи тільки до одного S-блоку, наприклад першого, відрізняються після етапу розширення.

Перший раунд в диференціалі проводиться завжди, а в другому раунді результати всіх S-блоків є рівними, крім першого S-блоку. На вході до третього раунду вхідні тексти для двох S-блоків (1 та 7) є завжди однаковими, оскільки перший S-блок не впливає на них згідно з таблицею 1.1. Тому виходи цих S-блоків є рівні, та результат XOR-у 8-ми бітів правої частини шифротекстів після трьох раундів є відомим, оскільки результат XOR-у виходів до другого раунду відомий. Праві частини після трьох раундів шифрування співпадають з лівими частинами після чотирьох раундів шифрування, саме тому результат XOR-у восьми бітів після чотирьох раундів відомий з імовірністю 1. Це буде використано для диференціальної атаки перших 6-ти раундів DES, яка використовує обмежену кількість відкритих текстів.

На рисунку 1.1 зображено процес пораундового утворення 4-раундового диференціала з вхідною різницею 20000000_x.

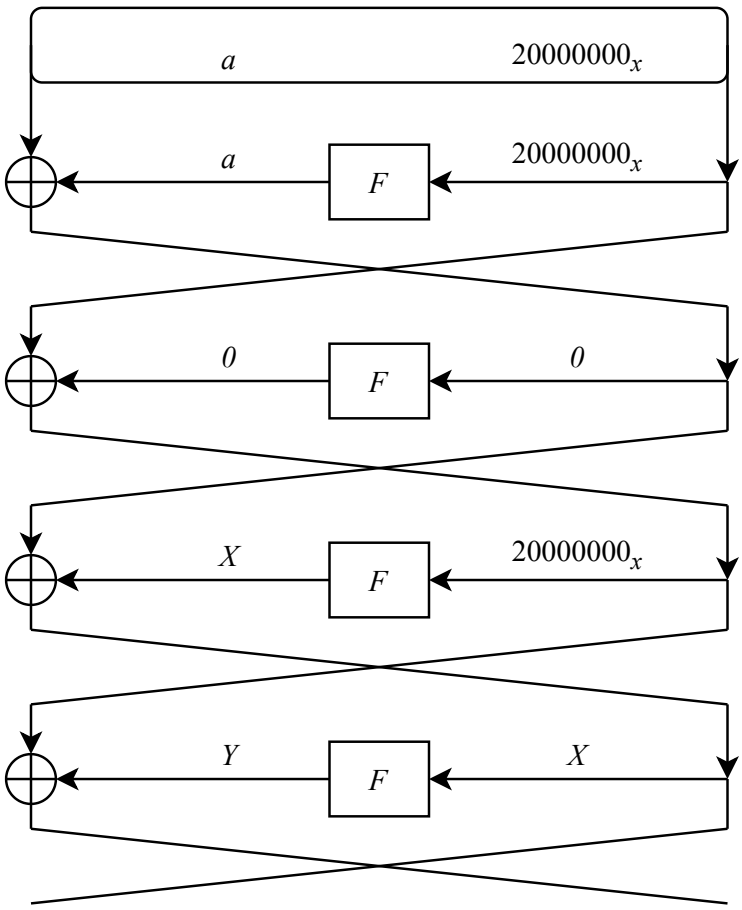

Рисунок 1.1 – 4 раундовий диференціал шифру DES

Теорема 1.2. *Існує диференціальна атака на* 6 *раундів* DES*, яка відновлює особистий ключ, використовуючи* 46 *вибраних відкритих текстів, за час* 3500 *шифрувань.*

Доведення. Розглянемо диференціальну атаку з вибраним відкритим текстом, використовуючи диференціал на рисунку 1.1 та аналогічний диференціал, де всі величини 20000000_x замінені на 40000000_x.

Спершу припустимо, що результати першого раунду шифрування мають різницю α. Входи третього раунду відрізняються лише 2 бітами та впливають лише на перший S-блок. Згідно з описом вище, входи з різницею X для четвертого раунду рівні входам до S-блоків 1 та 7. Тому 8 бітів різниці Y — нулі. Оскільки для криптоаналітика відома різниця входів третього раунду, зловмисник знає 8 бітів різниці виходів F-функції шостого раунду шифрування, тому що він знає різницю в шифротекстах.

Ці 8 бітів є вихідними бітами першого та сьомого S-блоків. Потім криптоаналітик, перебираючи всі можливі 64 значення ключа, перевіряє, чи входи в перший S-блок дають очікувану різницю вихідних текстів, і те ж саме для сьомого S-блоку.

Для кожної пари шифротекстів, що використовуються в аналізі, для обох S-блоків зловмисник отримає, в середньому, 4 можливих значення ключа, серед яких є правильні, оскільки такий диференціал має ймовірність 1. Спробувавши таке для декількох пар (наприклад 4 пари дають хорошу імовірність вибору ключа), зловмисник отримує правильне значення ключа, яке буде запропоновано всіма парами.

Більш детально дане доведення наведено в [2]. □

Приклад 1.2. *Атака на 5 раундів шифру* `Salsa20` [4]. В роботі Пауля Кроулі [5] описується диференціальна атака на основі усічених диференціалів на потоковий шифр `Salsa20`. Розглянемо детально як відбувається дана атака.

Для зручності позначимо `Salsa20-`w/r, де w — розмір слова, r — число раундів шифрування. Оригінальний шифр `Salsa20` позначається `Salsa20-32/20`. Слово — елемент $\mathbb{Z}/2^w\mathbb{Z}$. Визначимо бієктивну функцію S над вектором з чотирьох слів, параметризовану a, таким чином:

$$S_a((y_0, y_1, y_2, y_3)^T) = (y_1 \oplus ((y_0 + y_3) <<< a), y_2, y_3, y_0).$$

Композицію наступних функцій S об'єднаємо в одну функцію Q:

$$Q = S_{18} \circ S_{13} \circ S_9 \circ S_7.$$

Слід зауважити, що представлені константи застосовні для $w = 32$, вони можуть мінятися в залежності від w. Якщо виконати композицію цих

функцій по рядках, то отримаємо таку бієктивну функцію над матрицями:

$$Q'(m) = \begin{pmatrix} m_{1,1} & m_{1,2} & m_{1,3} & q_1 \\ m_{2,1} & m_{2,2} & m_{2,3} & q_2 \\ m_{3,1} & m_{3,2} & m_{3,3} & q_3 \\ m_{0,1} & m_{0,2} & m_{0,3} & q_0 \end{pmatrix},$$

$$\text{де } q = Q \begin{pmatrix} m_{0,0} \\ m_{1,0} \\ m_{2,0} \\ m_{3,0} \end{pmatrix} \text{ та } m = \begin{pmatrix} m_{0,0} & m_{0,1} & m_{0,2} & m_{0,3} \\ m_{1,0} & m_{1,1} & m_{1,2} & m_{1,3} \\ m_{2,0} & m_{2,1} & m_{2,2} & m_{2,3} \\ m_{3,0} & m_{3,1} & m_{3,2} & m_{3,3} \end{pmatrix}.$$

Використовуючи описану функцію Q, можна побудувати бієктивну функцію над матрицями слів розміру 4×4 таким чином:

$$R(m) = \left((Q')^4 (m) \right)^T.$$

Застосовуючи функцію R, визначається геш-функція шифру `Salsa20`:

$$H(m) = m + R^r(m).$$

Для виконання шифрування `Salsa20` необхідно ключ, який складається з восьми слів — $k_0, \ldots, k_7$ та двослівні параметри u, i: u_0, u_1 та i_0, i_1 відповідно; все це представляється у вигляді матриці, яка

складається з 16-ти слів таким чином:

$$Salsa20_k(u,i) = H\begin{pmatrix} c_0 & k_0 & k_1 & k_2 \\ k_3 & c_1 & v_0 & v_1 \\ i_0 & i_1 & c_2 & k_4 \\ k_5 & k_6 & k_7 & c_3 \end{pmatrix}$$

Тут $c_0, \ldots, c_3$ — константи, що залежать від довжини ключа, проте їхнє визначення опускаємо для спрощення.

Атака проводиться для 5 раундів шифру, тобто $r = 5$. Тоді $H(m) = m + R^5(m)$. Вісім з шістнадцяти комірок m для нас відомі, інша вісім комірок містять значення ключа. Якщо правильно відгадаємо k_3, це надасть нам можливість повністю знати рядок в $R^5(m)$, до якого можна застосувати Q^{-1}, щоб отримати один рядок $R^4(m)$. Якщо і далі продовжувати так робити, буде помітно, що якщо кожне вхідне слово, крім першого, для Q^{-1} відомо, то останнє вихідне слово може бути передбачено. Та якщо кожне вхідне слово, крім другого, відоме, то перше може бути передбачено. Якщо нам вдасться вгадати ключові слова $k_3, \ldots, k_7$, це нам надасть можливість передбачити такі комірки для $R^5(m)$:

$$\begin{pmatrix} \bullet & ? & ? & ? \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{pmatrix}$$

Якщо застосувати Q^{-1} до кожного рядка окрім першого, отримаємо

змогу передбачити такі вхідні комірки для $R^4(m)$:

$$\begin{pmatrix} ? & \bullet & \bullet & \bullet \\ ? & \bullet & \bullet & \bullet \\ ? & \bullet & \bullet & \bullet \\ ? & \bullet & \bullet & \bullet \end{pmatrix}$$

Якщо повторити такі процедуру, можемо передбачити такі вхідні комірки для $R^4(m)$:

$$\begin{pmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ \bullet & ? & ? & \bullet \end{pmatrix}$$

Обмеження, які застосовуються до бітів, ускладнює визначення слідів, які починаються з корисних комбінацій бітів; кожне слово, на яке опираємося (яке знаємо), поєднується з трьома словами, які не знаємо, перш ніж результати комбінуються між собою. Таким чином, наша вхідна різниця — це просто один біт у високому слові потокової позиції, вибраний для мінімізації нелінійного лавинного ефекту. Для вхідних текстів оберемо таку різницю:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0x80000000 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Після першого раунду (з імовірністю $1/2$) різниця буде виглядати так:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0x00201000 & ? & 0x80000000 & 0x00000100 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Після другого раунду (з імовірністю 2^{-9}) різниця матиме такий вигляд:

$$\begin{pmatrix} ? & 0x00201000 & 0x40200000 & 0x02000800 \\ ? & ? & ? & ? \\ ? & ? & ? & 0x00000040 \\ 0 & 0x00001000 & 0x00200000 & 0x04000080 \end{pmatrix}$$

Після третього раунду (з імовірністю 2^{-12}) різниця буде виглядати так:

$$\begin{pmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ 0x02002802 & ? & ? & ? \end{pmatrix}$$

Цей слід має достатньо високу ймовірність, щоб використовувати його як розпізнавач, з якого можна побудувати атаку. Однак можна зробити набагато краще. імовірність появи цієї різниці у виході набагато вища, ніж це вказано вище, фактично вона наближається до 2^{-9}. Це пояснюється тим, що існує багато інших диференціальних слідів з низькою вагою, які призводять до цієї різниці в комірці $R_3(m)_{3,0}$.

Розглядаючи багато слідів, можна побудувати набагато ефективнішу атаку. В роботі [5] наведено наступний приклад.

Експериментально визначимо набір з 1024 можливих різниць у комірці $R_3(m)_{3,0}$ від цієї вхідної різниці, так що ймовірність того, що один із них є правильним — приблизно 30%. Зі 32-ма вихідними парами, ймовірність того, що 5 або більше цих пар показують потрібну різницю, перевищує $1 - 2^{-3}$, тоді як ймовірність того, що цей поріг буде випадково досягнуто або перевищено — менше 2^{-99}. Було випробувано всі 2160 можливих значень для $k_3, \ldots, k_7$; для кожного, хто перевищує заданий поріг, пошук $k_0, \ldots, k_2$ відбувається шляхом простого перебору. Справжній ключ буде серед цих значень з імовірністю $1 - 2^{-3}$, і можна очікувати $2^{160-99} = 261$ помилкових позитивних результатів. Таким чином, важкість етапу пошуку буде приблизно $296 + 61 = 2157$, що значно менше, ніж важкість визначення кандидатів для $k_3, \ldots, k_7$.

Приклад 1.3. *Атака на шифр* `KLEIN` [6]. У роботі [7] представлено два нові усічені диференціальні шляхи для атаки шифру `KLEIN`, а також вдосконалений метод відновлення ключа шифрування. Дана атака застосовна до шифрів `KLEIN-64`, `KLEIN-80` та `KLEIN-96`.

Для шифру `KLEIN-64` атака знаходить ключ за допомогою 2^{58} операцій шифрування, використовуючи 2^{49} блоків пам'яті. Для шифру `KLEIN-80` із 14-ма раундами (замість стандартних 16-ти) атака знаходить ключ за допомогою 2^{78} операцій шифрування, використовуючи 2^{62} блоків пам'яті. Для шифру `KLEIN-96` із 15-ма раундами (замість стандартних 20-ти) атака знаходить ключ за допомогою 2^{92} операцій шифрування, використовуючи 2^{64} блоків пам'яті.

Приклад 1.4. *Атака на шифр* `RoadRunneR` [8]. У роботі [9] запропоновано атаку на легковаговий шифр `RoadRunneR` на основі методу «зустріч посередині» (англ. meet-in-the-middle). Атака застосовна до шифрів `RoadRunneR-80` та `RoadRunneR-128` зі зменшеною кількістю раундів. Для шифру `RoadRunneR-128` запропонована атака на основі усічених диференціалів на 7 раундів (замість стандартних 12-ти)

знаходить ключ за допомогою 2^{121} операцій шифрування, використовуючи 2^{68} блоків пам'яті та 2^{55} вибраних відкритих текстів.

Висновки до розділу 1

В цьому розділі розглянуто основні поняття диференціального криптоаналізу, досліджено історію виникнення поняття усіченого диференціала та наведено модель криптографічних атак на основі усічених диференціалів.

Для практичного ознайомлення з криптографічними атаками, які використовують усічені диференціали, досліджено відомі успішні атаки на такі шифри:

– шифр `DES` (атака дозволила зламати 6 раундів шифру);

– `Salsa20` (успішний злам 5 раундів шифру);

– шифри `KLEIN-64` та `KLEIN-80`, а також шифр `KLEIN-96` зі зменшеною кількістю раундів;

– шифри `RoadRunneR-80` та `RoadRunneR-128` зі зменшеною кількістю раундів.

Не зважаючи на відносно успішні випадки практичного застосування даного методу, на сьогодні не існує формальної теорії, що описує криптоаналіз на основі усічених диференціалів та дозволяє проводити оцінку стійкості конкретних шифрів до нього.

2 ФОРМАЛІЗАЦІЯ ТЕОРІЇ СТІЙКОСТІ ДО КРИПТОАНАЛІЗУ НА ОСНОВІ УСІЧЕНИХ ДИФЕРЕНЦІАЛІВ

В цьому розділі представлено формалізацію підходу до криптоаналізу на основі усічених диференціалів за схемою, аналогічною до класичного диференціального криптоаналізу. Для цього потрібно чітко визначити структуру усіченого диференціала. В даному розділі запропоновано дві різні інтерпретації структури усічених диференціалів. Для кожної інтерпретації пропонується параметр стійкості, що характеризує імовірність усіченого диференціала. Для цих параметрів наводяться результати їх практичної перевірки та доводяться деякі їхні характеристики й властивості.

2.1 Параметр стійкості для масок, які фіксують лише не змінені біти

У цьому розділі розглянуться такі усічені диференціали, які відслідковують лише не змінені біти так, як це було запропоновано Кнудсеном в [2]. Для цього введемо поняття усіченого диференціала і введемо параметр стійкості для S-блоку, який дозволить будувати нижні оцінки для імовірностей усічених диференціалів ітеративних блокових шифрів.

2.1.1 Формальне означення усіченого диференціала

Два вектори $\alpha, \beta \in V_n$ будемо трактувати як маски для вхідних та вихідних різниць. Принцип роботи цих масок наступний: якщо в масці на

деякій позиції стоїть 0, то і в різниці на цій самій позиції повинен бути 0; якщо в масці на деякій позиції стоїть 1, то в різниці на цій самій позиції може бути або 0, або 1. Таким чином, кожній масці у відповідність можна представити множину різниць, які можливі при цій масці. Визначимо множину можливих різниць для певної маски α таким чином:

$$\Delta(\alpha) = \{\alpha' \in V_n \setminus \{0\} : \alpha' \vee \alpha = \alpha\},$$

для нульової маски $\alpha = 0$ покладемо $\Delta(0) = 0$.

Розглянемо два вектори $\alpha, \beta \in V_n$ як маски вхідних та вихідних різниць, тоді пара (α, β) називається *усіченим диференціалом*. В даний інтерпретації усічені диференціали застосовуються для передбачення декількох нульових бітів вихідної різниці, що потенційно дозволяє будувати більш точні та потужні розпізнавачі для шифруючих перетворень (зокрема, блокових шифрів).

2.1.2 Параметр стійкості на основі безпосереднього означення

Безпосередньо з ідей усіченого диференціального криптоаналізу випливає, що для аналізу усічених диференціалів треба розглядати подію, що пара входів, різниця яких задовольняє заданій масці α, переходить у пару виходів, різниця яких відповідає заданій масці β. Відповідно, визначимо імовірність цієї події таким чином:

$$TDP^S(\alpha, \beta) = \frac{1}{2^n} \sum_{x \in V_n} [\exists \alpha' \in \Delta(\alpha), \exists \beta' \in \Delta(\beta) : S(x \oplus \alpha') = S(x) \oplus \beta'].$$

Дану імовірність, за аналогією до класичної імовірності диференціала, можна представити через таку множину двійкових

векторів у чисельнику:

$$TD^S(\alpha,\beta) = \{x \in V_n | \exists \alpha' \in \Delta(\alpha), \exists \beta' \in \Delta(\beta) : S(x \oplus \alpha') = S(x) \oplus \beta'\}.$$

Таким чином, отримуємо альтернативну формулу для обчислення імовірності усіченого диференціала:

$$TDP^S(\alpha,\beta) = 2^{-n}|TD^S(\alpha,\beta)|.$$

Введений параметр імовірності усіченого диференціала може бути використаний для пошуку високоймовірних усічених диференціалів марковських шифрів, який побудовано на використанні методу «гілок та границь» за аналогією до класичних диференціалів. Основна ідея цього алгоритму полягає в тому, що для заданої маски вхідної різниці послідовно шукаємо можливі маски вихідні різниці на кожному раунді, але ті усічені диференціали, імовірність яких є малою (тобто нижче встановленого порогового значення), відкидаємо. Таким чином, відбувається суттєва економія на обчислювальних ресурсах, оскільки розглядаємо не всі можливі шляхи поширення різниці поміж раундами, а лише гарантовано високоймовірні.

Обрахуємо значення цього параметру для 4-бітових S-блоків, запропонованих у роботі [11], а також слабких S-блоків, які відтворюють наступні функції: $\oplus$ вхідного тексту з певним числом та циклічний зсув вхідного тексту на декілька бітів. Таким чином отримаємо наступний список S-блоків:

- $K1 = (7, 9, 4, D, 0, 2, C, B, A, 8, 1, 6, E, 5, F, 3)$;
- $K2 = (1, 9, 6, 5, B, E, 2, 8, 4, A, F, 3, 0, 7, C, D)$;
- $K3 = (A, C, 3, 8, B, 7, D, 0, 4, 5, 1, F, E, 9, 6, 2)$;
- $K4 = (E, A, F, 1, 0, D, 7, 4, 5, 2, 8, 6, 3, B, 9, C)$;
- $K5 = (B, 0, D, E, 6, 4, 7, 9, 5, 1, C, 2, 8, F, A, 3)$;
- $K6 = (1, C, 3, 8, 0, 6, E, D, F, B, 4, 5, 9, 2, A, 7)$;
- $K7 = (E, 7, B, D, 8, 2, 5, 6, 3, 0, 1, C, F, 9, 4, A)$;

– $K8 = (4, 3, A, B, D, E, 8, 5, 9, 1, 7, C, F, 2, 6, 0)$;

– $xor\_5 : (5, 4, 7, 6, 1, 0, 3, 2, D, C, F, E, 9, 8, B, A)$;

– $xor\_7 : (7, 6, 5, 4, 3, 2, 1, 0, F, E, D, C, B, A, 9, 8)$;

– $xor\_11 : (B, A, 9, 8, F, E, D, C, 3, 2, 1, 0, 7, 6, 5, 4)$;

– $shift\_l\_2 : (0, 4, 8, C, 1, 5, 9, D, 2, 6, A, E, 3, 7, B, F)$;

– $shift\_l\_3 : (0, 8, 1, 9, 2, A, 3, B, 4, C, 5, D, 6, E, 7, F)$,

тут значення наведено в шістнадцятковій системі числення.

Результати розрахунків можна знайти за посиланням: https://bit.ly/2VV5UXj. Розрахунки показують, що цей параметр не адекватно характеризує імовірність усіченого диференціала та, відповідно, не може бути застосований для криптоаналізу.

Для ілюстрації вищесказаного розглянемо такий приклад. Візьмемо S-блок $K1$. В таблиці 2.1 наведено результати розрахунку потужності множини TD^S для усіх усічених диференціалів з $\alpha = 0111$ для вибраного S-блоку. Отже, імовірність певного усіченого диференціала можна обрахувати поділивши відповідну йому потужність множини TD^S на $2^4 = 16$. У таблиці значення маски β представлено в шістнадцятковій системі числення, проте надалі будемо використовувати двійкову.

Таблиця 2.1 – Результати обчислення потужності множини TD^S для усічених диференціалів виду $(0111, \beta)$ для обраного S-блоку

β	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\lvert TD^S(\alpha, \beta)\rvert$	0	4	8	16	8	14	12	16	4	14	16	16	14	16	16	16

Як можемо бачити, в таблиці є декілька варіантів β, при яких $TDP^S(0111_2, \beta) = 1$, наприклад, $\beta = A_{16} = 1010_2$ та $\beta = D_{16} = 1101_2$. Таким чином, маємо подію, що два тексти з маскою різниці 0111 на вході, після застосування до них S-блоку дають різницю з маскою 1010 на виході з імовірністю 1, тобто завжди. Така ж ситуація і з вихідною маскою різниці 1101. Можна стверджувати, що два тексти з маскою

різниці 0111 на вході, після застосування до них S-блоку, одночасно дають різницю з масками 1010 та 1101. Виходячи з наших позначень для масок, це можливо і породжує наступну подію: два тексти з маскою різниці 0111 на вході, після застосування до них S-блоку завжди дають різницю з маскою 1000 (ми порівняли побітово дві вихідні маски, і на місцях, де біти різні записали 0). Але це суперечить результатам наших розрахунків, оскільки з таблиці 2.1 можемо обрахувати імовірність усіченого диференціала $(0111, 1000)$ і вона дорівнює $\frac{1}{4}$.

Схожі ситуації є в розрахунках для кожного S-блоку з тих, які було розглянуто. Отже, не можна використовувати такий параметр для побудови розпізнавача шифруючого перетворення від випадкової функції.

2.1.3 Уточнений параметр стійкості для аналізу усічених диференціалів

Розвинемо нашу ідею для внесення адекватності в наш параметр оцінки імовірності усіченого диференціала, переформульовуючи його. Квантор $\exists$ біля різниці α' замінимо на квантор $\forall$, щоб для будь-якої вхідної різниці з множини $\Delta(\alpha)$ можна було гарантувати нулі на певних позиціях у вихідній масці з деякою імовірністю. Таким чином введемо нове означення для імовірності усіченого диференціала:

$$TDP^S(\alpha, \beta) = \frac{1}{2^n} \sum_{x \in V_n} [\forall \alpha' \in \Delta(\alpha), \exists \beta' \in \Delta(\beta) : S(x \oplus \alpha') = S(x) \oplus \beta'].$$

Тепер можемо змінити і означення множини TD:

$$TD^S(\alpha, \beta) = \{x \in V_n | \forall \alpha' \in \Delta(\alpha), \exists \beta' \in \Delta(\beta) : S(x \oplus \alpha') = S(x) \oplus \beta'\}.$$

Визначення імовірності усіченого диференціала через потужність множини TD залишається незмінним.

Наведемо деякі алгебраїчні властивості параметру TDP.

Лема 2.1. *При значенні параметра* $\alpha = 0$ *виконується таке співвідношення:*

$$TDP^S(0,\beta) = \begin{cases} 1, & \text{якщо } \beta = 0; \\ 0, & \text{інакше.} \end{cases}$$

Доведення. Для маски $\alpha = 0$ множина $\Delta(\alpha) = 0$, за означенням. Оскільки існує лише один диференціал з вхідною різницею 0, який має не нульову імовірність — це $(0,0)$. То для не нульової імовірності усіченого диференціала потрібно, щоб елемент 0 містився в множині $\Delta(\alpha)$. Виходячи з означення множини Δ, це можливо тільки коли $\beta = 0$. □

Лема 2.2. *Для параметру* $\beta = 11...1$ *та* $\forall\alpha \in V_n \setminus \{0\}$ *має місце таке співвідношення:*

$$TDP(\alpha, 11...1) = 1.$$

Доведення. Очевидно, що маска $\beta = 11...1$ покриває усі можливі вихідні різниці. Отже, для всіх можливих масок α, окрім нульової, вихідна різниця буде доступна в множині $\Delta(\alpha)$. □

Лема 2.3. *Для будь-яких значень параметрів* $\beta_1,\ \beta_2 \in V_n \setminus \{0\}$, *якщо* $\forall i$ *має місце* $\beta_{1_i} \geq \beta_{2_i}$, *то виконується співідношення*

$$\forall\alpha :\ TDP^S(\alpha,\beta_1) \geq TDP^S(\alpha,\beta_2),$$

причому $TD^S(\alpha,\beta_1) \supseteq TD^S(\alpha,\beta_2)$.

Доведення. Візьмемо довільні $\beta_1, \beta_2 \in V_n \setminus \{0\}$. Якщо $\forall i\ \beta_{1_i} \geq \beta_{2_i}$, тоді назвемо, що β_1 домінує над β_2, то це значить, що на місцях, на яких в β_2 стоять одиниці, в β_1 на цих місцях також стоять одиниці; на місцях, на яких в β_2 стоять нулі, в β_1 можуть бути як нулі так і одиниці. Це значить, що множина маска β_1 як мінімум покриває усі можливі різниці для маски β_2, тобто $\Delta(\beta_1) \supseteq \Delta(\beta_2)$. А це розширює можливості для будь-якої маски α. □

Наслідок 2.1. *Нехай* f *— лінійна функція відносно операції* $\oplus$ *і* β *домінує над* $f(\alpha)$, *тоді* $TDP^f(\alpha,\beta) = 1$.

Доведення. Оскільки функція f — лінійна відносно операції $\oplus$, тоді виконується:

$$DP^f(\alpha, \beta) = \big[f(x) \oplus \beta = f(x \oplus \alpha)\big] =$$

$$= \big[f(x) \oplus \beta = f(x) \oplus f(\alpha)\big] = \big[\beta = f(\alpha)\big].$$

Отже, для усічених диференціалів $\forall\alpha$ буде точно існувати одна маска $\beta = f(\alpha)$, для якої $TDP^f(\alpha, \beta) = 1$. Враховуючи лему 2.3: всі маски вихідних різниць, які домінують над β, будуть мати імовірність усіченого диференціала рівну 1 з даною маскою α. □

Обрахуємо значення цього параметру для S-блоків, згаданих у попередньому пункті. Результати розрахунків можна побачити за посиланням: https://bit.ly/2yguh8Q. Для цього випадку потрібно окремо розглядати S-блоки, запропонованими в роботі [11], назвемо їх «хорошими», та S-блоки утворені операцією $\oplus$ з деяким числом та циклічним зсувом входу на декілька бітів, назвемо їх «слабкими».

Спершу розглянемо слабкий S-блок, який відтворює операцію циклічного зсуву вліво на два біти. Як показано в таблиці 2.2, для всіх β, окрім 1100, 1101, 1110, 1111, імовірність усіченого диференціала дорівнює нулю. Проте для $\beta = 1100$ імовірність усіченого диференціала дорівнює 1. Слід зауважити, що маска 1100 утворюється з маски 0011 шляхом циклічного зсуву вліво на два біти, що і є операцією, яку симулює вибраний слабкий S-блок. Інші маски для β, при яких імовірність усіченого диференціала дорівнює 1, містять маску 1100, відповідно до наших позначень.

Таблиця 2.2 – Результати обчислення TDP^S для усічених диференціалів виду $(0011, \beta)$ для обраного слабкого S-блоку

β	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$TDP^S(\alpha, \beta)$	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Така ситуація справедлива для всіх слабких S-блоків. Для будь-якої маски вхідної різниці α^*, якщо до неї застосувати функцію, яку симулює певний слабкий S-блок, отримаємо маску вихідної різниці β^*, для якої імовірність усіченого диференціала з вибраним раніше α^* дорівнює 1. Звичайно, імовірність усіченого диференціала для вибраного α^* зі всіма вихідними масками, які містять маску β^*, також дорівнює 1. Що ілюструє наслідок 2.1.

Для будь-якого з хороших S-блоків ($K1 - K8$) отримуємо наступну ситуацію. Для всіх α, для яких потужність множини можливих вхідних різниць $\Delta(\alpha)$ більше одиниці, тобто квантор, який був замінений, вступає в дію, імовірність всіх усічених диференціалів, окрім $\beta = 1111$, дорівнює нулю, що продемонстровано в таблиці 2.3. Виокремлення $\beta = 1111$, пояснюється тим, що не залежно від вибору вхідної різниці, імовірність отримати будь-яку (саме це означає маска 1111) вихідну різницю завжди буде рівна 1. Варто зауважити, що нульове значення TDP не означає, що відповідні диференціали неможливі, оскільки TDP розглядає усі диференціали, які відповідають заданим маскам у сукупності.

Таблиця 2.3 – Результати обчислення TDP^S для усічених диференціалів виду $(0011, \beta)$ для обраного хорошого S-блоку

β	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$TDP^S(\alpha, \beta)$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Для α, для яких потужність множини $\Delta(\alpha) = 1$, множина $TD^S(\alpha, \beta)$ буде складатися з усіх векторів, які, маючи різницю α на вході, породжують хоча б одну з різниць з множини $\Delta(\beta)$. Для таких випадків імовірність усіченого диференціала варіюється від 0 до $\frac{1}{2}$, не враховуючи вихідну маску 1111. У таблиці 2.4 ми можемо бачити приклад описаного випадку для $\alpha = 0100$.

Таблиця 2.4 – Результати обчислення TDP^S для усічених диференціалів виду $(0100, \beta)$ для обраного хорошого S-блоку

β	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$TDP^S(\alpha, \beta)$	0	0	0	0	0	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{2}$	0	$\frac{1}{8}$	0	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{2}$	$\frac{1}{2}$	1

Описана вище поведінка нашого параметру не є неадекватною. Проте, значення цього параметру в ситуаціях, які представляють найбільшу цікавість — $|\Delta(\alpha)| > 1$, не є релевантними. Це може пояснюватися малим розміром простору, на якому відбувається пошук вхідних векторів для множини TD. Тому обрахуємо значення цього параметру для 8-бітових S-блоків зі специфікації шифру `Калина` [12]. Для кожного з запропонованих S-блоків побудуємо таблицю, де для усіх можливих пар масок $(\alpha, \beta),\ \alpha, \beta \in V_8$, буде обраховано:

– $DP^S(\alpha, \beta)$,

– $\max_{\alpha' \in \Delta(\alpha), \beta' \in \Delta(\beta)} DP^S(\alpha', \beta')$,

– $TDP^S(\alpha, \beta)$,

– $TD^S(\alpha, \beta)$.

Результати обчислення можна знайти за посиланням: https://bit.ly/2z9xhnu.

Перше, що видно з результатів — це те, що для $\alpha = 0$ результати усіх $TDP^S(0, \beta) = 0$, що є очікувано, оскільки нульова різниця вхідних текстів може породити лише нульову різницю вихідних текстів, яка не покривається жодною з масок. Наступною тривіальною властивістю є те, що для будь-якого α при $\beta = 11111111$ імовірність усіченого диференціала (α, β) дорівнює одиниці, що також є очікувано, оскільки будь-яка різниця вхідних текстів може породити будь-яку різницю вихідних текстів.

Якщо розглянути окремо α такі, що $|\Delta(\alpha)| = 1$, то «новий» підхід ніяк не відрізняється від підходу, описаного в попередньому розділі, оскільки в цьому випадку квантор $\forall$ не надає ніяких додаткових

обмежень. В такому випадку множина $TD^S(\alpha, \beta)$ буде складатися з усіх векторів, які, маючи різницю α на вході, породжують хоча б одну з різниць з множини $\Delta(\beta)$. Слід зауважити, що усі α для яких $|\Delta(\alpha)| = 1$, характеризуються тим, що у них вага — 1. Для таких випадків

$$TDP^S(\alpha, \beta) \in \left[\frac{2}{2^8}, \frac{148}{2^8}\right].$$

В більшості з усіх інших випадків не описаних вище, $TDP^S(\alpha, \beta) = 0$, проте розглянемо випадки коли це не так. Для цих випадків, коли імовірність усіченого диференціала не дорівнює нулю, характерною ознакою є вага маски α, яка рівна або 2, або 3, та має місце правило «домінуючої» β. Розглянемо дію цього правила на одному з таких усічених диференціалів. Як бачимо з результатів в таблиці 2.5, для маски β, в якої зафіксовано три нульових елементи, отримуємо множину можливих відкритих текстів потужності 4, яка складається з таких елементів: 70_{16}, 71_{16}, 72_{16}, 73_{16}. Для β з двома фіксованими нульовими елементами, на місцях, що співпадають з нульовими елементами попередньої β, множина $TD^S(\alpha, \beta)$ містить вісім елементів, чотири з яких співпадають з елементами множини відкритих текстів для $\beta = 00011111_2$. Враховуючи наші позначення, можемо стверджувати, що маска 00111111_2 домінує над маскою 00011111_2. Оскільки

$$TDP^S(03_{16}, 3F_{16}) = \frac{8}{2^8};$$

$$TDP^S(03_{16}, 1F_{16}) = \frac{4}{2^8},$$

то маємо

$$TDP^S(03_{16}, 3F_{16}) > TDP^S(03_{16}, 1F_{16}).$$

Таким чином, отримаємо таке співвідношення

$$TD^S(03_{16}, 1F_{16}) \subset TD^S(03_{16}, 3F_{16}),$$

що ілюструє лему 2.3.

Таблиця 2.5 – Частина результатів розрахунку параметрів для S-блоку π_0 зі специфікації шифру `Калина`, для маски $\alpha = 00000011$

β	00011111	00111111	11011110
$\lvert D^S(\alpha,\beta)\rvert$	2	2	0
$\max DP^S(\alpha',\beta')$	4	4	6
$\lvert TD^S(\alpha,\beta)\rvert$	4	8	4
$TD^S(\alpha,\beta)$	$[70_{16}, 71_{16}, 72_{16}, 73_{16}]$	$[04_{16}, 05_{16}, 06_{16}, 07_{16}, 70_{16}, 71_{16}, 72_{16}, 73_{16}]$	$[132_{16}, 133_{16}, 134_{16}, 135_{16}]$

Також слід зауважити, що в будь-якому випадку, коли $TDP^S(\alpha,\beta) > 0$, маємо:

$$TDP^S(\alpha,\beta) > DP^S(\alpha,\beta).$$

При цьому отримати співвідношення величин

$$TDP^S(\alpha,\beta) \quad \text{та} \quad \max_{\alpha'\in\Delta(\alpha),\ \beta'\in\Delta(\beta)} DP^S(\alpha',\beta')$$

не можливо, оскільки, за результатами розрахунків, не виявлено ніякої залежності між цими величинами, це продемонстровано в таблиці 2.5.

У випадках, коли даний метод дає ненульову нижню оцінку імовірності усіченого диференціала, вона виходить більшою за імовірність звичайного диференціала, що потенційно дозволяє будувати більш ефективні атаки.

Отже, у даному підрозділі було запропоновано два параметри стійкості, які характеризують імовірність усіченого диференціала. Один з

яких спростовано, та показано, що він не інформативний. Для іншого параметру, побудованого на основі попереднього, наведено деякі характеристики та властивості та показано, що він може використовуватись для побудови атак на основі усічених диференціалів. Слід зауважити, що дана ідея формалізації теорії стійкості до криптоаналізу на основі усічених диференціалів не є виключною та єдиною можливою.

2.2 Модифікований вигляд усіченого диференціала

В цьому розділі розглянемо такі усічені диференціали, які відслідковують не лише не змінені біти, але й біти, які обов'язково будуть зміненими. Для цього заново введемо поняття усіченого дифренціалу і введемо параметр стійкості для S-блоку. Також буде обраховано значення цього параметру для 4- та 8-бітових S-блоків.

2.2.1 Формальне означення розширеного усіченого диференціала

Розглянемо простір усіх можливих формальних векторів довжини n з елементами з множини $\{0, 1, *\}$, який позначимо як V_n^*. Два вектори $\alpha, \beta \in V_n^*$ будемо трактувати як маски для вхідних та вихідних різниць: якщо в масці на деякій позиції стоїть 0, то і в різниці на відповідній позиції стоїть 0, якщо 1 – то і в різниці 1; якщо ж в масці на певній позиції стоїть $*$, то в різниці на цій позиції може бути як 0, так і 1 (нам не важливо, чи змінилось щось на заданій позиції). Кожній масці $\alpha \in V_n^*$ у відповідність можна представити множину $\Delta(\alpha)$ усіх різниць $\alpha' \in V_n$, які відповідають даній масці. *Усіченим диференціалом* буде називатися пара векторів (α, β).

Визначимо імовірність переходу від вхідної маски α до вихідної маски

β таким чином:

$$TDP^S(\alpha,\beta) = \frac{1}{2^n}\sum_{x\in V_n}\left[\ \forall\alpha'\in\Delta(\alpha), \exists\beta'\in\Delta(\beta) : S(x\oplus\alpha') = S(x)\oplus\beta'\ \right].$$

Дану імовірність, за аналогією до класичної імовірності диференціала, можна представити через таку множину двійкових векторів, які відповідають ненульовим індикаторам у чисельнику:

$$TD^S(\alpha,\beta) = \left\{x\in V_n \mid \forall\alpha'\in\Delta(\alpha), \exists\beta'\in\Delta(\beta) : S(x\oplus\alpha') = S(x)\oplus\beta'\right\}.$$

Наведемо деякі алгебраїчні властивості параметру TDP для усічених диференціалів на просторі V_n^*.

Лема 2.4. *Якщо $\beta\in\{0,*\}^n$, то $TDP^S(0,\beta) = 1$, для всіх інших β виконується співвідношення $TDP^S(0,\beta) = 0$.*

Доведення. Ця властивість випливає з наявності тривіального диференціала $(0,0)$, який завжди має імовірність 1. А оскільки в масці β відсутні елементи 1, то в множину $\Delta(\beta)$ включається нульовий елемент. А в множині $\Delta(0)$ є лише один елемент — нуль. Отже, в множину $TD^S(0,\beta)$ попадуть усі елементи простору V_n. □

Лема 2.5. *Для усіх значень параметру $\alpha\in V_n^*$ виконується:*

$$TDP^S(\alpha, **\cdots*) = 1.$$

Доведення. Очевидно, що маска $**\cdots*$ накриває усі можливі маски, тому для довільного α та довільного входу x вихідна різниця точно буде відповідати одній з різниць, що накриваються маскою $**\cdots*$. □

Лема 2.6. *Якщо $\alpha\in\{0,1\}^n$, то виконується:*

$$TDP^S(\alpha,\beta) \geq \max_{\alpha'\in\Delta(\alpha),\beta'\in\Delta(\beta)} DP^S(\alpha',\beta').$$

Доведення. Якщо $\alpha \in \{0,1\}^n$, то множина різниць, що відповідають масці α, тобто $\Delta(\alpha)$, складається з одного вектору, а маска α і є єдиною можливою вхідною різницею. При цьому кількість можливих вихідних різниць не зменшується. Якщо ж $\alpha, \beta \in \{0,1\}^n$, то маємо

$$TDP^S(\alpha, \beta) = DP^S(\alpha, \beta),$$

оскільки в цьому випадку усічений диференціал збігається з класичним. □

Для цього означення маски введемо поняття домінації. Будемо стверджувати, що маска α_1 доміную над маскою α_2, якщо $\forall i$ або $\alpha_{1_i} = \alpha_{2_i}$, або $\alpha_{1_i} = *$, а $\alpha_{2_i} \in \{0,1\}$. Тоді маска α_1 буде покривати усі різниці доступні масці α_2.

Лема 2.7. $\forall$ β_1, $\beta_2 \in V_n^*$, *якщо* β_1 *домінує над* β_2, *то* $\forall \alpha$ $TDP^S(\alpha, \beta_1) \geq TDP^S(\alpha, \beta_2)$, *причому* $TD^S(\alpha, \beta_1) \supseteq TD^S(\alpha, \beta_2)$.

Доведення. Візьмемо довільні $\beta_1, \beta_2 \in V_n \setminus \{0\}$. Якщо β_1 домінує над β_{2_i}, то це значить, що маска β_1 як мінімум покриває усі можливі різниці для маски β_2, тобто $\Delta(\beta_1) \supseteq \Delta(\beta_2)$. А це розширює можливості для будь-якої маски α. □

2.2.2 Опис результатів експериментальних обчислень для запропонованого параметра

Для початку, як і для попередньої формалізації усічених диференціалів, обчислимо значення параметру для 4-бітових S-блоків. Повні результати обрахунків можна знайти в таблицях за посиланням: https://bit.ly/2YwA450.

Проілюструємо поведінку введеного параметру на прикладі S-блоку $K1$ з [11], який показано в попередньому пункті.

Для випадків, коли $|\Delta(\alpha)| > 3$, усі значення $TDP^S(\alpha, \beta)$ виявились

нульовими; це можна пояснити тим, що умова виконання події в TDP дуже строга для S-блоків невеликого розміру. Для випадків, коли потужність множини $\Delta(\alpha)$ дорівнює 2 або 3, імовірність усічених диференціалів приймає значення від 0 до $0,5$; для більшості з них $TDP^S(\alpha, \beta) = 0$, однак є достатньо диференціалів із ненульовими TDP. Як можна побачити з таблиці 2.6, встановити пряму залежність між значенням TDP та імовірностями звичайних диференціалів, які відповідають заданим усіченим, доволі важко: TDP може бути як більше, так і менше, а іноді навіть точно співпадає зі значенням імовірності диференціала. Це свідчить, в першу чергу, про те, що шифри із гарантованою стійкістю проти класичного диференціального криптоаналізу можуть виявитись нестійкими до атак на основі усічених диференціалів в межах запропонованого формалізованого методу.

Таблиця 2.6 – Значення параметрів для деяких усічених диференціалів S-блоку $K1$

$(\alpha,\ \beta)$	$\max DP^S(\alpha', \beta')$	$TDP^S(\alpha, \beta)$	$TD^S(\alpha, \beta)$
$(001*, 10*1)$	$0,125$	$0,125$	{0111, 1010}
$(001*, 11**)$	$0,25$	$0,125$	{0110, 1011}
$(001*, 1***)$	$0,25$	$0,5$	{0100, 0101, 0110, 0111 1000, 1001, 1010, 1011}
$(001*, *101)$	$0,25$	0	$\varnothing$
$(001*, *10*)$	$0,25$	$0,0625$	{0001}

З таблиці 2.6 на прикладі усічених диференціалів $(001*, 10*1)$ та $(001*, 1***)$ бачимо ілюстрацію леми 2.7, оскільки маска $1***$ домінує над маскою $10*1$ та мають місце такі співвідношення:

$$TDP^S(001*, 1***) > TDP^S(001*, 10*1);$$

$$TD^S(001*, 1***) \supseteq TD^S(001*, 10*1).$$

Таблиця 2.7 – Значення параметрів для деяких усічених диференціалів S-блоку $K1$ з особливою маскою вхідної різниці

$(\alpha,\ \beta)$	$\max DP^S(\alpha', \beta')$	$TDP^S(\alpha, \beta)$	$TD^S(\alpha, \beta)$
$(0001, 1011)$	$0,125$	$0,125$	$\{1100,\ 1101\}$
$(0001, 101*)$	$0,125$	$0,125$	$\{1100,\ 1101\}$
$(0001, 10*1)$	$0,125$	$0,25$	$\{0010, 0011, 1100, 1101\}$
$(0001, 10**)$	$0,125$	$0,25$	$\{0010, 0011, 1100, 1101\}$

У таблиці 2.7 можемо бачити ілюстрацію леми 2.6. В цьому випадку має $\alpha = 0001$, і для усіх β виконується:

$$TDP^S(\alpha, \beta) \geq \max_{\alpha' \in \Delta(\alpha),\ \beta' \in \Delta(\beta)} DP^S(\alpha', \beta').$$

Наприклад, при значенні $\beta = 1011$ маємо:

$$TDP^S(0001, 1011) = \max_{\alpha' \in \Delta(0001),\ \beta' \in \Delta(1011)} DP^S(\alpha', \beta'),$$

а при $\beta = 10*1$ маємо:

$$TDP^S(0001, 10*1) > \max_{\alpha' \in \Delta(0001),\ \beta' \in \Delta(10*1)} DP^S(\alpha', \beta'),$$

Результати обчислення введеного нами параметру для 8-бітових

S-блоків зі специфікації шифру `Калина` можна знайти за посиланням: https://bit.ly/3bdiKVy. Для 8-бітових S-блоків переважна частина усічених диференціалів має імовірність нуль, проте всі усічені диференціали, які мають не нульову імовірність ілюструють всі вище наведені леми 2.4 — 2.7. Також слід зауважити, що в загальному випадку не можна побудувати будь-якої залежності між значеннями

$$TDP^S(\alpha, \beta) \quad \text{та} \quad \max_{\alpha' \in \Delta(\alpha), \beta' \in \Delta(\beta)} DP^S(\alpha', \beta').$$

Оскільки і для 4-бітових і для 8-бітових S-блоків велика частина усічених диференціалів має нульову імовірність, в таблицях 2.8 та 2.9 наведено порівняння кількості не нульових усічених диференціалів для кожного з S-блоків, для яких проводились обрахунки. Кількість усіх можливих 4-бітових усічених диференціалів дорівнює $3^4 \cdot 3^4 = 6561$, а кількість усіх можливих 8-бітових усічених диференціалів — $3^8 \cdot 3^8 = 43046721$.

Таблиця 2.8 – Порівняння кількості ненульових усічених диференціалів для 4-бітових S-блоків

S-блок	Кількість ненульових усічених диференціалів	Відношення
$K1$	1615	0.246
$K2$	1515	0.23
$K3$	1615	0.246
$K4$	1698	0.259
$K5$	1698	0.259
$K6$	1611	0.245
$K7$	1515	0.23
$K8$	1698	0.259

Таблиця 2.9 – Порівняння кількості ненульових усічених диференціалів для 8-бітових S-блоків

S-блок	Кількість ненульових усічених диференціалів	Відношення
π_0	3398813	0.079
π_1	3400294	0.079
π_2	3425164	0.08
π_3	346486	0.08
π_0^{-1}	3372999	0.078
π_1^{-1}	3459818	0.08
π_2^{-1}	3440409	0.08
π_3^{-1}	3483927	0.081

Якщо порівняти результати в таблицях 2.8 та 2.9, то бачимо, що ненульових усічених диференціалів для 8-бітових S-блоків, які були розглянуті, в середньому більше ніж в 2000 разів більше ніж для 4-бітових S-блоків, які були розглянуті. Проте у відсотковому співвідношенні їх в середньому в 3 рази менше.

Висновки до розділу 2

В даному розділі запропоновано формалізований підхід до побудови усічених диференціалів на основі шаблонів, які відслідковують лише не змінені біти.

На основі цього підходу проведено спробу запропонувати параметр стійкості, який характеризує імовірність диференціала. Запропоновано два параметри, один з яких спростовано шляхом доведення його неінформативності для аналізу імовірності диференціала. Для іншого

параметра, що побудований на основі попереднього, доведено деякі його властивості та характеристики, та, таким чином, обґрунтовано доцільність його застосування для побудови атак на основі усічених диференціалів.

Також запропоновано формалізований підхід до побудови усічених диференціалів на основі шаблонів, які, окрім відслідковування не змінених бітів, розглядають також обов'язкові зміни у різницях текстів. Для цього підходу також запропоновано параметр стійкості для S-блоків, що характеризує імовірність усіченого диференціала.

3 ПЕРЕВІРКА ЗАПРОПОНОВАНИХ МЕТОДІВ ОЦІНЮВАННЯ НА МОДЕЛЬНОМУ ШИФРІ

У даному розділі наводяться необхідні відомості про модельний шифр, який використовується для перевірки запропонованих методів оцінювання імовірностей усічених диференціалів. Також покроково описується проведений експеримент пошуку високоімовірнісних усічених диференціалів для даного модельного шифру та представляються результати побудованого експерименту.

3.1 Опис модельного шифру

В ролі модельного шифру будемо розглядати шифр Хейса [13]. Шифр Хейса — це ітеративний блоковий шифр, що побудований на структурі SP-мережі. Шифр Хейса був одним з перших шифрів, для якого автори, Г. Хейс та С. Таварес, намагались теоретично довести стійкість до диференціального криптоаналізу; в подальшому виявилось, що розроблена ними теорія не гарантувала захищеності від даного типу атак. Однак шифр виявився зручним для пояснення ідей диференціального криптоаналізу.

На рисунку 3.1 зображено один раунд шифру Хейса з параметром $n = 8$, тобто розмір вхідного блоку дорівнює $n^2 = 64$. Ключ K_i — раундовий ключ шифру. Слід зауважити, що всі S-блоки однакові.

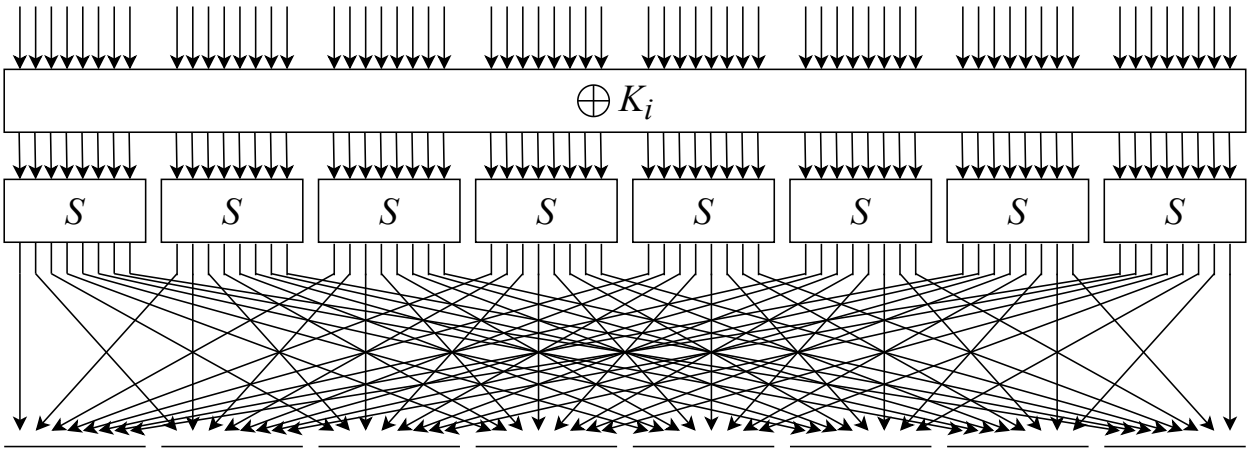

Рисунок 3.1 – Один раунд шифру Хейса при $n = 8$

Шифр Хейса складається із раундів $F_k(x)$, $x \in V_n$ — вхідний блок, що перетворюють блоки розміром n^2 біт. Раунд складається із таких кроків (рис. 3.1):

1) додавання вхідного блоку з раундовим ключем: $y = x \oplus k$;

2) розбиття отриманого блоку на фрагменти по n бітів кожен:

$$y = (y_0,\ y_1, \ldots, y_{n-1});$$

3) перетворення кожного фрагменту розбиття за допомогою n-бітових S-блоків:

$$z = \big(S(y_0),\ S(y_1), \ldots S(y_{n-1})\big);$$

4) перестановка бітів отриманого блоку, яка відбувається таким чином: i-тий біт j-того блоку стає j-тим бітом i-того блоку.

В даній роботі використовується шифр Хейса з параметром $n = 8$.

Лема 3.1. *Якщо* $\forall x \in V_n,\ \forall k \in V_n$ *має місце* $f_k(x) = S(x \oplus k)$, *то виконується:*

$$\forall k \in v_n,\ \forall \alpha \in v_n,\ \forall \beta \in v_n :\ TDP^{f_k}(\alpha, \beta) = TDP^S(\alpha, \beta).$$

Доведення. За означенням:

$$TDP^{f_k}(\alpha,\beta) = \frac{1}{2^n}\sum_{x\in V_n}[\forall\alpha'\in\Delta(\alpha),\exists\beta'\in\Delta(\beta) : f_k(x\oplus\alpha') = f_k(x)\oplus\beta'].$$

Оскільки $f_k(x) = S(x\oplus k)$, то:

$$TDP^{f_k}(\alpha,\beta) = \\ = \frac{1}{2^n}\sum_{x\in V_n}[\forall\alpha'\in\Delta(\alpha),\exists\beta'\in\Delta(\beta) : S(x\oplus\alpha'\oplus k) = S(x\oplus k)\oplus\beta'].$$

Значення k — фіксоване, а значення x проходить по всіх множині V_n, то значення $y = x\oplus k$ також буде проходити по всіх множині V_n. Отже, можемо зробити таку заміну:

$$TDP^{f_k}(\alpha,\beta) = \frac{1}{2^n}\sum_{y\in V_n}[\forall\alpha'\in\Delta(\alpha),\exists\beta'\in\Delta(\beta) : S(y\oplus\alpha') = S(y)\oplus\beta'].$$

У правій частині рівності отримали означення $TDP^S(\alpha,\beta)$. Отже, $TDP^{f_k}(\alpha,\beta) = TDP^S(\alpha,\beta)$, що і необхідно було довести. □

3.2 Постановка і опис експерименту

Метою нашого експерименту є показати, що запропонований нами метод оцінювання усічених диференціалів може бути застосовний на практиці. Завдання цього експерименту полягає в тому, щоб, використовуючи метод «гілок і границь», знайти усічений диференціал для якомога більшої кількості раундів, імовірність якого відрізняється від імовірності при випадковому виборі усіченого диференціала.

Для досягнення цілі експерименту було запропоновано алгоритм пошуку високоімовірнісних усічених диференціалів, який побудовано на основі методу «гілок і границь». Основна ідея цього алгоритму полягає в тому, що для заданої маски вхідної різниці послідовно шукаємо можливі маски вихідних різниць на кожному раунді, але ті різниці, імовірність

яких є малою (тобто нижче встановленого порогового значення), відкидаємо. Недоліком такого підходу (як і при будь-якому евристичному пошуку) є те, що можемо відкинути практично всі можливі шляхи як варіанти із малою імовірністю і, таким чином, алгоритм не знайде нічого.

З розрахунків, які проводились для даної роботи і описані в попередньому розділі, нам відомі ймовірності усіх усічених диференціалів для 8-бітових S-блоків зі специфікації шифру `Калина`. Тому для нашого модельного шифру будемо використовувати один з розглянутих S-блоків, а саме π_0^{-1}, для кожного блоку, на які розбивається вхідний блок.

Оскільки цей алгоритм застосовується до шифру Хейса з параметром $n = 8$, то вхідна маска буде складатися з восьми 8-бітових масок. Маска вхідної різниці $\alpha \in V_{64}$ розбивається на вісім масок

$$\alpha_0,\ \alpha_1, \ldots \alpha_7 \in V_8.$$

Тому можемо визначити формулу обчислення імовірності усіченого диференціала для шару застосування S-блоків, тобто $S = S_0|S_1|\ldots|S_7$, таким чином:

$$\forall \alpha, \beta \in V_{64} : TDP^S(\alpha, \beta) = \\ = TDP^{S_0}(\alpha_0, \beta_0) \cdot TDP^{S_1}(\alpha_1, \beta_1) \cdot \ldots \cdot TDP^{S_7}(\alpha_7, \beta_7), \quad (3.1)$$

де $\forall i = 0, \ldots, 7$: $\alpha_i, \beta_i \in V_8$.

Оскільки перемішування бітів в четвертому кроці раунду шифру Хейса є лінійною операцією, тому це ніяк не впливає на значення імовірності усіченого диференціала, проте сам усічений диференціал зазнає змін. Таким чином, якщо лінійне перетворення позначити функцією $l(x)$, то можемо стверджувати, що виконується таке співвідношення:

$$\forall \alpha,\ \beta \in V_{64} :\ TDP^S(\alpha, \beta) = TDP^{S\,|\,l}(\alpha, l(\beta)).$$

Враховуючи лему 3.1, можемо упустити крок додавання ключа до вхідного блоку. З розрахунків, які проводились для даної роботи і описані в попередньому розділі, нам відомі ймовірності усіх усічених диференціалів для 8-бітових S-блоків зі специфікації шифру `Калина`. Тому для нашого модельного шифру будемо використовувати один з розглянутих S-блоків, а саме π_0^{-1}.

Отже, алгоритм пошуку високоімовірнісних диференціалів для певної маски α за один раунд буде виглядає таким чином.

1) Маска вхідної різниці $\alpha \in V_n$ розбивається на вісім 8-бітових масок

$$\alpha_0, \alpha_1, \ldots \alpha_7.$$

2) Для кожної маски $\alpha_i, i = 0, \ldots, 7$ будується множина можливих масок вихідних текстів для вибраного S-блоку, таких, щоб імовірність усіченого диференціала була більшою від деякого параметра q, тобто

$$\Gamma(\alpha_i) = \beta_i : TDP^S(\alpha_i, \beta_i) > q.$$

Таким чином, отримаємо вісім таких множин.

3) Будуємо декартів добуток усіх множин, що були отримані на попередньому раунді і отримуємо вектори з восьми 8-бітових масок. Отримані вектори і будуть 64-бітовими масками вихідних різниць для шару застосування S-блоків. Таким чином, отримано всі можливі маски вихідних різниць для маски вхідної різниці α.

4) Використовуючи формулу 3.1 обчислюється значення імовірності кожного можливого усіченого диференціала з α. На цьому етапі відбувається ще одне відсіювання: всі усічені диференціали, імовірність яких менша від певного параметра p — відкидаються.

5) До всіх масок вихідних різниць, що пройшли попередній пункт, застосовується лінійне перемішування бітів.

Таким чином, після виконання наведених кроків, отримаємо множину можливих масок вихідних різниць з певною імовірністю (яка

зберігається від попереднього раунду) для маски вхідних різниць α.

Далі до кожної з отриманих масок вихідних різниць β першого раунду застосовуємо такий самий алгоритм, тільки на четвертому кроці обчислена імовірність усічених диференціалів множаться на значення усіченого диференціала для β, отриманого з першого раунду. це і буде значенням усіченого диференціала після двох і більше раундів шифрування.

Якщо виникає ситуація, що для двох різних результатів якогось раунду після наступних раундів отримується однакова маска вихідної різниці, то їхні імовірності додаються.

Від значень q та p залежить точність та швидкість алгоритму. Якщо порогові значення великі, то алгоритм буде відкидати значну кількість диференціалів (і працювати швидше), однак успішність пошуку буде падати. При низьких значеннях цих параметрів алгоритм обробляє значно більше можливих шляхів, тому точність пошуку зростає, але робота уповільнюється, а зберігання списків вимагатиме значно більше пам'яті.

3.3 Результати експерименту та їх інтерпретація

В ході виконання дослідження створено програмну реалізацію побудованого алгоритму пошуку високоімовірнісних усічених диференціалів. Текст створеної програми наведено в додатку 3.3.

Експеримент проводився на комп'ютері з такими характеристиками:

1) процесор: `Quad-Core Intel Core i5` 8-го покоління;

2) частота процесора: 2.3 ГГц;

3) об'єм оперативної пам'яті: 8 ГБ.

Перші спроби запустити алгоритм зі значеннями параметрів $q = 0$ та $p = 10^{-5} \approx 2^{-16.6}$ на всі 8 раундів шифру Хейса завершились невдачою, оскільки така реалізація алгоритму потребувала дуже багато пам'яті.

Тому для продовження дослідження емпіричним методом було

знайдено оптимальні значення параметрів для обмеження імовірностей для комп'ютера на якому проводились обчислення, а саме такі значення параметрів: $q = 0.0001$ та $p = 0.00001$.

В ролі маски вхідної різниці було обрано маску з одним ненульовим наймолодшим бітом. При розбитті маски вхідної різниці на вісім 8-бітових масок отримаємо:

– сім перших масок нульовими;

– остання маска має вигляд 00000001.

Таке значення останньої маски вхідної різниці було обрано для того, щоб розтягнути якомога далі поширення одиничних бітів, щоб побудувати диференціал для якнайбільшої кількості раундів.

Максимальна кількість раундів, для якої дана реалізація алгоритму змогла розрахувати значення імовірностей усічених диференціалів, — це три раунди.

У таблиці 3.1 наведено декілька прикладів отриманих масок вихідної різниці для вхідної різниці $[0, 0, 0, 0, 0, 0, 0, 1]$. Запис маски у такому вигляді одразу розділяє маску вхідної різниці на вісім блоків. Відповідно кожне число визначає маску певного 8-бітового блоку і записане у десятковому вигляді. Якщо це число перевести у двійковий вигляд, то отримаємо звичне представлення маски і наглядно будемо бачити які біти залишаються незмінними, а які можуть змінитися. Вихідні різниці у таблиці 3.1 записані за таким же принципом.

Таблиця 3.1 – Частина результатів пошуку високоімовірнісних усічених диференціалів для модельного шифру Хейса

Маска вихідної різниці	Імовірність	Імовірність у вигляді 2^{-x}
[0,1,1,5,5,5,5,5]	$1.67 \cdot 10^{-5}$	$2^{-15.87}$
[2,0,0,2,0,2,2,2]	$6.62 \cdot 10^{-5}$	$2^{-13.88}$
[2,0,0,2,2,0,2,2]	$6.62 \cdot 10^{-5}$	$2^{-13.88}$
[0,2,0,2,2,2,2,2]	$13.23 \cdot 10^{-5}$	$2^{-12.88}$

Розглянемо один з наведених прикладів детальніше. Наприклад, усічений диференціал виду

$$\big([0,0,0,0,0,0,0,1],\ [2,0,0,2,2,0,2,2]\big)$$

має імовірність $6.62{\cdot}10^{-5}$ після 3 раундів шифрування. Це означає, що якщо у вхідному тексті не міняти усі біти, окрім останнього, то з імовірністю $6.62{\cdot}10^{-5}$ на виході отримаємо не змінені біти на всіх місцях, окрім других з кінця бітів першого, четвертого, п'ятого, сьомого та восьмого блоків. Таким чином, ці п'ять бітів можуть бути змінені після трьох раундів шифрування завдяки одній можливій зміні у вхідному тексті.

Значення імовірності диференціала насправді показує нижню міру імовірності для реальних різниць. Враховуючи наші позначення, на місцях, де в масках стоїть 1, в реальних різницях може бути як 1 так і 0, тобто значення цього біту може бути зміненим. Оскільки в означенні імовірності усіченого диференціала перевіряється виконання умови для всіх вхідних різниць, які відповідають певній масці, то для кожної з них це значення буде рівним, або меншим справжньому.

Висновки до розділу 3

В даному розділі наведено алгоритм пошуку високоімовірнісних усічених диференціалів, що оснований на методі «гілок і границь» та використовує у своїй будові параметр стійкості TDP, який характеризує імовірність усіченого диференціала.

Розділ містить результати застосування побудованого алгоритму до модельного шифру Хейса при використанні 8-бітового S-блоку π_3^{-1} зі специфікації шифру `Калина`. Як наслідок, отримано декілька високоімовірнісних усічених диференціалів для трьох раундів шифрування даного модельного шифру. Для отримання усічених диференціалів для більшої кількості раундів шифрування необхідно використовувати алгоритм на комп'ютері з більшою кількістю оперативної пам'яті.

ВИСНОВКИ

При виконанні даної дисертаційної роботи проведено аналіз опублікованих досліджень ефективності криптографічного аналізу на основі усічених диференціалів. В результаті аналізу встановлено, що існує декілька прикладів успішного застосування атак на основі усічених диференціалів до сучасних шифрів або їх модифікацій, однак, не зважаючи на відносно успішні випадки практичного застосування даного методу, на сьогодні не існує формальної теорії, що описує криптоаналіз на основі усічених диференціалів та дозволяє проводити оцінку стійкості конкретних шифрів до нього.

В роботі запропоновано формалізований підхід побудови усічених диференціалів на основі шаблонів, які відслідковують лише не змінені в наслідок процедури шифрування біти. Використовуючи такий підхід побудови, виконано дослідження можливих параметрів стійкості, що характеризують імовірність усіченого диференціала. Як наслідок, в роботі наведено два параметри стійкості. Доведено, що один з цих параметрів стійкості є неінформативний для оцінки ймовірності усіченого диференціала. Інший параметр стійкості побудований на основі першого, однак може бути застосовним для побудови атак на основі усічених диференціалів, що також доводиться у роботі шляхом визначення характеристик та необхідних властивостей даного параметра стійкості.

Запропонований параметр стійкості використаний в алгоритмі для пошуку високоімовірнісних усічених диференціалів у модельному шифрі. Проведено експеримент зі застосуванням запропонованого алгоритму. Даний експеримент закінчився успішно, було знайдено декілька високоімовірнісних усічених диференціали для модельного шфиру з неповною кількістю раундів.

У даній кваліфікаційній роботі також запропоновано інший формалізований підхід до побудови усічених диференціалів. При цьому

підході використовуються шаблони, що відслідковують не лише незмінні при шифруванні біти, а й зміни у різницях текстів. Запропоновано параметр стійкості для S-блоків, який дозволяє проводити побудову усічених диференціалів з гарантованою імовірністю їх появи. Показано, що для конкретного S-блоку значення запропонованого параметру слабко пов'язані з імовірностями диференціалів, тому стійкість шифру до класичного диференціального криптоаналізу не може свідчити про стійкість до криптографічного аналізу на основі усічених диференціалів.

У подальшому розвитку дослідження планується виконати уточнення та систематизацію підходів побудови атак на основі усічених диференціалів, а також моделей та методів їх застосування до конкретних схем шифрування; дослідити ефективність побудованого алгоритму пошуку високоімовірнісних усічених диференціалів, та, якщо можливо, удосконалити його.

ПЕРЕЛІК ПОСИЛАНЬ

1. Biham E. Differential cryptanalysis of DES-like cryptosystems / E. Biham, A. Shamir // Journal of Cryptology. — 1991. — №4. — С. 3 – 72.

2. Knudsen L. Truncated and Higher Order Differentials / Lars R. Knudsen // Fast Software Encryption: Second International Workshop. Proceedings / Lars R. Knudsen. — Leuven, Belgium: Springer, 1994. — (Lecture Notes in Computer Science; вип. 1008). — С. 196 – 211.

3. National Bureau of Standards. Data encryption standard / National Bureau of Standards, U.S. Department of Commerce. // Federal Information Processing Standard (FIPS). — 1977. — №46.

4. Bernstein D. J. The Salsa20 family of stream ciphers [Електронний ресурс] / Daniel J. Bernstein // 2007. — Режим доступу до ресурсу: https://cr.yp.to/snuffle/salsafamily-20071225.pdf.

5. Crowley P. Truncated differential cryptanalysis of five rounds of Salsa20 [Електронний ресурс] / Paul Crowley // IACR Cryptology ePrint Archive, Report 2005/375. — 2005. — Режим доступу до ресурсу: https://eprint.iacr.org/2005/375.pdf.

6. KLEIN: A New Family of Lightweight Block Ciphers [Електронний ресурс] / G. Zheng, N. Svetla, L. Yee Wei // Centre for Telematics and Information Technology (CTIT) — 2010. — Режим доступу до ресурсу: https://research.utwente.nl/files/5095831/The_KLEIN_Block_Cipher.pdf.

7. An Improved Truncated Differential Cryptanalysis of KLEIN [Електронний ресурс] / S.Rasoolzadeh, Z. Ahmadian, M. Salmasizadeh, M. Reza Aref // IACR Cryptology ePrint Archive, Report 2014/485. — 2014. — Режим доступу до ресурсу: https://eprint.iacr.org/2014/485.pdf.

8. Baysal A. RoadRunneR: A Small and Fast Bitslice Block Cipher for Low Cost 8-Bit Processors [Електронний ресурс] / Adnan Baysal, Suhap Sahin // IACR Cryptology ePrint Archive, Report 2015/906. — 2015. — Режим доступу до ресурсу: https://eprint.iacr.org/2015/906.pdf.

9. Truncated Differential Analysis of Round-Reduced RoadRunneR Block Cipher [Електронний ресурс] / Q.Yang, L. Hu, S. Sun, L. Song // IACR Cryptology ePrint Archive, Report 2016/084. — 2016. — Режим доступу до ресурсу: https://eprint.iacr.org/2016/084.pdf.

10. Courtois N. An Improved Differential Attack on Full GOST [Електронний ресурс] / Nicolas T. Courtois // IACR Cryptology ePrint Archive, Report 2012/138. — 2012. — Режим доступу до ресурсу: https://eprint.iacr.org/2012/138.pdf.

11. Яковлєв С. В. Збалансовані критерії якості довгострокових ключових елементів алгоритму шифрування ГОСТ 28147-89 / Сергій Володимирович Яковлєв. // Інформаційні технології та комп'ютерна інженерія. — 2009. — №1(14) — С. 48 – 55. — ISSN 1999-9941.

12. Oliynykov R. A New Encryption Standard of Ukraine: The Kalyna Block Cipher [Електронний ресурс] / R. Oliynykov, I. Gorbenko, O. Kazymyrov // IACR Cryptology ePrint Archive, Report 2015/650. — 2015. — Режим доступу до ресурсу: https://eprint.iacr.org/2015/650.pdf.

13. Heys Howard M. A Tutorial on Linear and Differential Cryptanalysis [Електронний ресурс] / Howard M. Heys. — 2002. —- Режим доступу : http://www.engr.mun.ca/~howard/PAPERS/ldc_tutorial.pdf

ДОДАТОК А ТЕКСТИ ПРОГРАМ

Далі представлено тексти програм для реалізації обчислення імовірностей усічених диференціалів та інших параметрів для різних S-блоків.

А.1 Програмна реалізація обрахунку імовірностей усічених диференціалів та інших параметрів

```
import itertools
import time
import pandas as pd
from s_box import S_box
from extended_binary_mask import binary_string, extended_binary_mask, create_delta_set

def create_table(s_box_n_bit, V_n, save=False):
  D_table = dict()
  for alpha in V_n:
  temp_x_list_alpha = set()
  for beta in V_n:
  temp_x_list = set()
  for i in V_n:
  s_box_result = s_box_n_bit.calculate_result(i)
  s_box_alpha_result = s_box_n_bit.calculate_result(i ^ alpha)
  if s_box_alpha_result == s_box_result ^ beta:
    temp_x_list.add(i)
  D_table[(alpha, beta)] = temp_x_list

  # TD_table = dict()
  # max_DP_table = dict()
  # delta_alpha_table = dict()
  # delta_beta_table = dict()
  result = dict()
  # with open('res.csv', 'w') as file:
  i = 0
  for pair in alpha_beta_pairs:
  i+=1
  if i % 1000 == 0:
  print(pair)
  delta_alpha = precalculated_delta_set[pair[0]]
```

```
delta_beta = precalculated_delta_set[pair[1]]
all_diff = itertools.product(delta_alpha, delta_beta)
temp_x_list = list()
temp_max_dp = list()
# print(bool(delta_alpha and delta_beta))
if delta_alpha and delta_beta:
for x in V_n:
# for x in list(possible_x):
flag = True
for alpha_1 in delta_alpha:
  # print((s_box_n_bit.calculate_result(x ^ alpha_1) ^ s_box_n_bit.calculate_result(x)))
  if (s_box_n_bit.calculate_result(x ^ alpha_1) ^
   s_box_n_bit.calculate_result(x)) not in delta_beta:
  flag = False
  break
if flag:
  temp_x_list.append(x)
for differential in all_diff:
temp_max_dp.append(len(D_table[differential]))
# print(temp_x_list)
if temp_x_list:
# line = '{},{},{},{},{},{}'.format(pair, max(temp_max_dp)
#  if temp_max_dp else 0, len(temp_x_list), len(delta_alpha))
# file.write(line)
# file.write('\n')
# print(1)
# TD_table[pair] = temp_x_list
# max_DP_table[pair] = max(temp_max_dp) if temp_max_dp else 0
# delta_alpha_table[pair] = delta_alpha
# delta_beta_table[pair] = delta_beta
result[pair] = [max(temp_max_dp) if temp_max_dp else 0]
result[pair].append(len(temp_x_list))
result[pair].append(temp_x_list)
result[pair].append(delta_alpha)
result[pair].append(delta_beta)

# result = dict()
# for i in TD_table:
# if len(delta_alpha_table[i]) > 5 and i[1]!='****':
# result[i] = [max_DP_table[i]]
# result[i].append(len(TD_table[i]))
# result[i].append(TD_table[i])
# result[i].append(delta_alpha_table[i])
# result[i].append(delta_beta_table[i])
```

```
  df = pd.DataFrame.from_dict(data=result,
  orient='index', columns=[
   "maxDP(a',␣b')",
   "TDP(a,␣b)",
   'TD(a,b)',
   'delta(alpha)',
   'delta(beta)'])

  if save:
  if s_box_n_bit.name:
  df.to_csv(s_box_n_bit.name + '.csv')
  print('File␣was␣saved!')
  else:
  raise Exception('File␣cannot␣be␣saved:␣no␣S-box␣name!')
  return df

# start
mask_elements = ['0', '1', '*']
n = 8
V_n = list(range(2 ** n))

time_start = time.time()
all_masks = [''.join(i) for i in list(itertools.product(mask_elements, repeat=n))]
print('All␣masks␣built.')
# print(len(all_masks))
print("Time:␣{}".format(time.time() - time_start))

# calculate all possible alpha-beta pairs
time_start = time.time()
alpha_beta_pairs = list(itertools.product(all_masks, repeat=2))
# for alpha in all_masks:
#   for beta in all_masks:
#   alpha_beta_pairs.append(tuple([alpha, beta]))
print('All␣possible␣alpha-beta␣pairs␣calculated.')
print(len(alpha_beta_pairs))
# print(alpha_beta_pairs[:10])
print('Time:␣{}'.format(time.time() - time_start))

#precalculate delta for each mask
time_start = time.time()
precalculated_delta_set = dict()
for mask in all_masks:
  precalculated_delta_set[mask] = create_delta_set(extended_binary_mask(mask))
print('Delta␣sets␣precalculated.')
```

```
# print(precalculated_delta_set['0101**11'])
print('Time: {}'.format(time.time() - time_start))


s_box_dict = {
  #   'kalyna_1': [168, 67, 95, 6, 107, 117, 108, 89, 113, 223, 135, 149,
  23, 240, 216, 9, 109, 243, 29, 203, 201, 77, 44, 175, 121, 224, 151, 253,
  111, 75, 69, 57, 62, 221, 163, 79, 180, 182, 154, 14, 31, 191, 21, 225, 73,
  210, 147, 198, 146, 114, 158, 97, 209, 99, 250, 238, 244, 25, 213, 173, 88,
  164, 187, 161, 220, 242, 131, 55, 66, 228, 122, 50, 156, 204, 171, 74, 143,
  110, 4, 39, 46, 231, 226, 90, 150, 22, 35, 43, 194, 101, 102, 15, 188, 169,
  71, 65, 52, 72, 252, 183, 106, 136, 165, 83, 134, 249, 91, 219, 56, 123,
  195, 30, 34, 51, 36, 40, 54, 199, 178, 59, 142, 119, 186, 245, 20, 159, 8,
  85, 155, 76, 254, 96, 92, 218, 24, 70, 205, 125, 33, 176, 63, 27, 137, 255,
  235, 132, 105, 58, 157, 215, 211, 112, 103, 64, 181, 222, 93, 48, 145, 177,
  120, 17, 1, 229, 0, 104, 152, 160, 197, 2, 166, 116, 45, 11, 162, 118, 179,
  190, 206, 189, 174, 233, 138, 49, 28, 236, 241, 153, 148, 170, 246, 38, 47,
  239, 232, 140, 53, 3, 212, 127, 251, 5, 193, 94, 144, 32, 61, 130, 247, 234,
  10, 13, 126, 248, 80, 26, 196, 7, 87, 184, 60, 98, 227, 200, 172, 82, 100,
  16, 208, 217, 19, 12, 18, 41, 81, 185, 207, 214, 115, 141, 129, 84, 192, 237,
  78, 68, 167, 42, 133, 37, 230, 202, 124, 139, 86, 128]
  #   'kalyna_2': [206, 187, 235, 146, 234, 203, 19, 193, 233, 58, 214, 178,
  210, 144, 23, 248, 66, 21, 86, 180, 101, 28, 136, 67, 197, 92, 54, 186, 245,
  87, 103, 141, 49, 246, 100, 88, 158, 244, 34, 170, 117, 15, 2, 177, 223, 109,
  115, 77, 124, 38, 46, 247, 8, 93, 68, 62, 159, 20, 200, 174, 84, 16, 216, 188,
  26, 107, 105, 243, 189, 51, 171, 250, 209, 155, 104, 78, 22, 149, 145, 238,
  76, 99, 142, 91, 204, 60, 25, 161, 129, 73, 123, 217, 111, 55, 96, 202, 231,
  43, 72, 253, 150, 69, 252, 65, 18, 13, 121, 229, 137, 140, 227, 32, 48, 220,
  183, 108, 74, 181, 63, 151, 212, 98, 45, 6, 164, 165, 131, 95, 42, 218, 201,
  0, 126, 162, 85, 191, 17, 213, 156, 207, 14, 10, 61, 81, 125, 147, 27, 254,
  196, 71, 9, 134, 11, 143, 157, 106, 7, 185, 176, 152, 24, 50, 113, 75, 239,
  59, 112, 160, 228, 64, 255, 195, 169, 230, 120, 249, 139, 70, 128, 30, 56, 225,
  184, 168, 224, 12, 35, 118, 29, 37, 36, 5, 241, 110, 148, 40, 154, 132, 232,
  163, 79, 119, 211, 133, 226, 82, 242, 130, 80, 122, 47, 116, 83, 179, 97, 175,
  57, 53, 222, 205, 31, 153, 172, 173, 114, 44, 221, 208, 135, 190, 94, 166, 236,
  4, 198, 3, 52, 251, 219, 89, 182, 194, 1, 240, 90, 237, 167, 102, 33, 127, 138,
  39, 199, 192, 41, 215]
  #   'kalyna_3': [147, 217, 154, 181, 152, 34, 69, 252, 186, 106, 223, 2, 159,
  220, 81, 89, 74, 23, 43, 194, 148, 244, 187, 163, 98, 228, 113, 212, 205, 112,
  22, 225, 73, 60, 192, 216, 92, 155, 173, 133, 83, 161, 122, 200, 45, 224, 209,
  114, 166, 44, 196, 227, 118, 120, 183, 180, 9, 59, 14, 65, 76, 222, 178, 144,
  37, 165, 215, 3, 17, 0, 195, 46, 146, 239, 78, 18, 157, 125, 203, 53, 16, 213,
  79, 158, 77, 169, 85, 198, 208, 123, 24, 151, 211, 54, 230, 72, 86, 129, 143,
  119, 204, 156, 185, 226, 172, 184, 47, 21, 164, 124, 218, 56, 30, 11, 5, 214,
  20, 110, 108, 126, 102, 253, 177, 229, 96, 175, 94, 51, 135, 201, 240, 93, 109,
  63, 136, 141, 199, 247, 29, 233, 236, 237, 128, 41, 39, 207, 153, 168, 80, 15,
```

```
55, 36, 40, 48, 149, 210, 62, 91, 64, 131, 179, 105, 87, 31, 7, 28, 138, 188,
32, 235, 206, 142, 171, 238, 49, 162, 115, 249, 202, 58, 26, 251, 13, 193, 254,
250, 242, 111, 189, 150, 221, 67, 82, 182, 8, 243, 174, 190, 25, 137, 50, 38,
176, 234, 75, 100, 132, 130, 107, 245, 121, 191, 1, 95, 117, 99, 27, 35, 61, 104,
42, 101, 232, 145, 246, 255, 19, 88, 241, 71, 10, 127, 197, 167, 231, 97, 90, 6,
70, 68, 66, 4, 160, 219, 57, 134, 84, 170, 140, 52, 33, 139, 248, 12, 116, 103]
#   'kalyna_4': [104, 141, 202, 77, 115, 75, 78, 42, 212, 82, 38, 179, 84, 30,
25, 31, 34, 3, 70, 61, 45, 74, 83, 131, 19, 138, 183, 213, 37, 121, 245, 189, 88,
47, 13, 2, 237, 81, 158, 17, 242, 62, 85, 94, 209, 22, 60, 102, 112, 93, 243, 69,
64, 204, 232, 148, 86, 8, 206, 26, 58, 210, 225, 223, 181, 56, 110, 14, 229, 244,
249, 134, 233, 79, 214, 133, 35, 207, 50, 153, 49, 20, 174, 238, 200, 72, 211,
48, 161, 146, 65, 177, 24, 196, 44, 113, 114, 68, 21, 253, 55, 190, 95, 170, 155,
136, 216, 171, 137, 156, 250, 96, 234, 188, 98, 12, 36, 166, 168, 236, 103, 32,
219, 124, 40, 221, 172, 91, 52, 126, 16, 241, 123, 143, 99, 160, 5, 154, 67, 119,
33, 191, 39, 9, 195, 159, 182, 215, 41, 194, 235, 192, 164, 139, 140, 29, 251,
255, 193, 178, 151, 46, 248, 101, 246, 117, 7, 4, 73, 51, 228, 217, 185, 208, 66,
199, 108, 144, 0, 142, 111, 80, 1, 197, 218, 71, 63, 205, 105, 162, 226, 122,
167, 198, 147, 15, 10, 6, 230, 43, 150, 163, 28, 175, 106, 18, 132, 57, 231, 176,
130, 247, 254, 157, 135, 92, 129, 53, 222, 180, 165, 252, 128, 239, 203, 187,
107, 118, 186, 90, 125, 120, 11, 149, 227, 173, 116, 152, 59, 54, 100, 109, 220,
240, 89, 169, 76, 23, 127, 145, 184, 201, 87, 27, 224, 97]
#   'kalyna_5': [164, 162, 169, 197, 78, 201, 3, 217, 126, 15, 210, 173, 231,
211, 39, 91, 227, 161, 232, 230, 124, 42, 85, 12, 134, 57, 215, 141, 184, 18,
111, 40, 205, 138, 112, 86, 114, 249, 191, 79, 115, 233, 247, 87, 22, 172, 80,
192, 157, 183, 71, 113, 96, 196, 116, 67, 108, 31, 147, 119, 220, 206, 32, 140,
153, 95, 68, 1, 245, 30, 135, 94, 97, 44, 75, 29, 129, 21, 244, 35, 214, 234,
225, 103, 241, 127, 254, 218, 60, 7, 83, 106, 132, 156, 203, 2, 131, 51, 221,
53, 226, 89, 90, 152, 165, 146, 100, 4, 6, 16, 77, 28, 151, 8, 49, 238, 171, 5,
175, 121, 160, 24, 70, 109, 252, 137, 212, 199, 255, 240, 207, 66, 145, 248, 104,
10, 101, 142, 182, 253, 195, 239, 120, 76, 204, 158, 48, 46, 188, 11, 84, 26, 166,
187, 38, 128, 72, 148, 50, 125, 167, 63, 174, 34, 61, 102, 170, 246, 0, 93, 189,
74, 224, 59, 180, 23, 139, 159, 118, 176, 36, 154, 37, 99, 219, 235, 122, 62, 92,
179, 177, 41, 242, 202, 88, 110, 216, 168, 47, 117, 223, 20, 251, 19, 73, 136, 178,
236, 228, 52, 45, 150, 198, 58, 237, 149, 14, 229, 133, 107, 64, 33, 155, 9, 25,
43, 82, 222, 69, 163, 250, 81, 194, 181, 209, 144, 185, 243, 55, 193, 13, 186, 65,
17, 56, 123, 190, 208, 213, 105, 54, 200, 98, 27, 130, 143]
#   'kalyna_6': [131, 242, 42, 235, 233, 191, 123, 156, 52, 150, 141, 152, 185,
105, 140, 41, 61, 136, 104, 6, 57, 17, 76, 14, 160, 86, 64, 146, 21, 188, 179,
220, 111, 248, 38, 186, 190, 189, 49, 251, 195, 254, 128, 97, 225, 122, 50, 210,
112, 32, 161, 69, 236, 217, 26, 93, 180, 216, 9, 165, 85, 142, 55, 118, 169, 103,
16, 23, 54, 101, 177, 149, 98, 89, 116, 163, 80, 47, 75, 200, 208, 143, 205, 212,
60, 134, 18, 29, 35, 239, 244, 83, 25, 53, 230, 127, 94, 214, 121, 81, 34, 20, 247,
30, 74, 66, 155, 65, 115, 45, 193, 92, 166, 162, 224, 46, 211, 40, 187, 201, 174,
106, 209, 90, 48, 144, 132, 249, 178, 88, 207, 126, 197, 203, 151, 228, 22, 108,
250, 176, 109, 31, 82, 153, 13, 78, 3, 145, 194, 77, 100, 119, 159, 221, 196, 73,
```

```
138, 154, 36, 56, 167, 87, 133, 199, 124, 125, 231, 246, 183, 172, 39, 70, 222,
223, 59, 215, 158, 43, 11, 213, 19, 117, 240, 114, 182, 157, 27, 1, 63, 68, 229,
135, 253, 7, 241, 171, 148, 24, 234, 252, 58, 130, 95, 5, 84, 219, 0, 139, 227, 72,
12, 202, 120, 137, 10, 255, 62, 91, 129, 238, 113, 226, 218, 44, 184, 181, 204,
110, 168, 107, 173, 96, 198, 8, 4, 2, 232, 245, 79, 164, 243, 192, 206, 67, 37, 28,
33, 51, 15, 175, 71, 237, 102, 99, 147, 170]
#   'kalyna_7': [69, 212, 11, 67, 241, 114, 237, 164, 194, 56, 230, 113, 253,
182, 58, 149, 80, 68, 75, 226, 116, 107, 30, 17, 90, 198, 180, 216, 165, 138,
112, 163, 168, 250, 5, 217, 151, 64, 201, 144, 152, 143, 220, 18, 49, 44, 71,
106, 153, 174, 200, 127, 249, 79, 93, 150, 111, 244, 179, 57, 33, 218, 156, 133,
158, 59, 240, 191, 239, 6, 238, 229, 95, 32, 16, 204, 60, 84, 74, 82, 148, 14,
192, 40, 246, 86, 96, 162, 227, 15, 236, 157, 36, 131, 126, 213, 124, 235, 24,
215, 205, 221, 120, 255, 219, 161, 9, 208, 118, 132, 117, 187, 29, 26, 47, 176,
254, 214, 52, 99, 53, 210, 42, 89, 109, 77, 119, 231, 142, 97, 207, 159, 206, 39,
245, 128, 134, 199, 166, 251, 248, 135, 171, 98, 63, 223, 72, 0, 20, 154, 189, 91,
4, 146, 2, 37, 101, 76, 83, 12, 242, 41, 175, 23, 108, 65, 48, 233, 147, 85, 247,
172, 104, 38, 196, 125, 202, 122, 62, 160, 55, 3, 193, 54, 105, 102, 8, 22, 167,
188, 197, 211, 34, 183, 19, 70, 50, 232, 87, 136, 43, 129, 178, 78, 100, 28, 170,
145, 88, 46, 155, 92, 27, 81, 115, 66, 35, 1, 110, 243, 13, 190, 61, 10, 45, 31,
103, 51, 25, 123, 94, 234, 222, 139, 203, 169, 140, 141, 173, 73, 130, 228, 186,
195, 21, 209, 224, 137, 252, 177, 185, 181, 7, 121, 184, 225]
#   'kalyna_8': [178, 182, 35, 17, 167, 136, 197, 166, 57, 143, 196, 232, 115,
34, 67, 195, 130, 39, 205, 24, 81, 98, 45, 247, 92, 14, 59, 253, 202, 155, 13,
15, 121, 140, 16, 76, 116, 28, 10, 142, 124, 148, 7, 199, 94, 20, 161, 33, 87,
80, 78, 169, 128, 217, 239, 100, 65, 207, 60, 238, 46, 19, 41, 186, 52, 90, 174,
138, 97, 51, 18, 185, 85, 168, 21, 5, 246, 3, 6, 73, 181, 37, 9, 22, 12, 42, 56,
252, 32, 244, 229, 127, 215, 49, 43, 102, 111, 255, 114, 134, 240, 163, 47, 120,
0, 188, 204, 226, 176, 241, 66, 180, 48, 95, 96, 4, 236, 165, 227, 139, 231, 29,
191, 132, 123, 230, 129, 248, 222, 216, 210, 23, 206, 75, 71, 214, 105, 108, 25,
153, 154, 1, 179, 133, 177, 249, 89, 194, 55, 233, 200, 160, 237, 79, 137, 104,
109, 213, 38, 145, 135, 88, 189, 201, 152, 220, 117, 192, 118, 245, 103, 107,
126, 235, 82, 203, 209, 91, 159, 11, 219, 64, 146, 26, 250, 172, 228, 225, 113,
31, 101, 141, 151, 158, 149, 144, 93, 183, 193, 175, 84, 251, 2, 224, 53, 187,
58, 77, 173, 44, 61, 86, 8, 27, 74, 147, 106, 171, 184, 122, 242, 125, 218, 63,
254, 62, 190, 234, 170, 68, 198, 208, 54, 72, 112, 150, 119, 36, 83, 223, 243,
131, 40, 50, 69, 30, 164, 211, 162, 70, 110, 156, 221, 99, 212, 157]
'k1': [7, 9, 4, 13, 0, 2, 12, 11, 10, 8, 1, 6, 14, 5, 15, 3],
'k2': [1, 9, 6, 5, 11, 14, 2, 8, 4, 10, 15, 3, 0, 7, 12, 13],
'k3': [10, 12, 3, 8, 11, 7, 13, 0, 4, 5, 1, 15, 14, 9, 6, 2],
'k4': [14, 10, 15, 1, 0, 13, 7, 4, 5, 2, 8, 6, 3, 11, 9, 12],
'k5': [11, 0, 13, 14, 6, 4, 7, 9, 5, 1, 12, 2, 8, 15, 10, 3],
'k6': [1, 12, 3, 8, 0, 6, 14, 13, 15, 11, 4, 5, 9, 2, 10, 7],
'k7': [14, 7, 11, 13, 8, 2, 5, 6, 3, 0, 1, 12, 15, 9, 4, 10],
'k8': [4, 3, 10, 11, 13, 14, 8, 5, 9, 1, 7, 12, 15, 2, 6, 0],
'xor_5': [5, 4, 7, 6, 1, 0, 3, 2, 13, 12, 15, 14, 9, 8, 11, 10],
```

```
  'xor_7': [7, 6, 5, 4, 3, 2, 1, 0, 15, 14, 13, 12, 11, 10, 9, 8],
  'xor_11': [11, 10, 9, 8, 15, 14, 13, 12, 3, 2, 1, 0, 7, 6, 5, 4],
  'shift_left_2': [0, 4, 8, 12, 1, 5, 9, 13, 2, 6, 10, 14, 3, 7, 11, 15],
  'shift_left_3': [0, 8, 1, 9, 2, 10, 3, 11, 4, 12, 5, 13, 6, 14, 7, 15]
  }

for s_box_name in s_box_dict:
  time_start = time.time()
  curent_s_box = S_box(n, name=s_box_name, s_box=s_box_dict[s_box_name])
  a = create_table(curent_s_box, V_n, save=True)
  print('Time: {}'.format(time.time() - time_start))
```

A.2 Програмна реалізація симуляції роботи S-блоку

```
class S_box:

 def __init__(self, bits, name=None, s_box=None):
  if s_box and len(s_box) == 2 ** bits:
   self.bits = bits
   self.s_box = s_box

  else:
   if bits == 2:
    self.bits = 2
    self.s_box = [2, 1, 3, 0]

   if bits == 3:
    self.bits = 3
    self.s_box = [1, 6, 5, 4, 0, 7, 3, 2]

   if bits == 4:
    self.bits = 4
    self.s_box = [7, 9, 4, 13, 0, 2, 12, 11, 10, 8, 1, 6, 14, 5, 15, 3]

  self.name = name

 def calculate_result(self, input_number):
  '''
   input must be a positive integer
  '''
  if input_number.bit_length() > self.bits:
   raise Exception('Input bit length > S_box bit length!')
  else:
   return self.s_box[input_number]
```

А.3 Програмна реалізація роботи розширених усічених диференціалів

```
import re

class binary_string:
  def __init__(self, value, length):
  if type(value) == type(1):
    bin_val = bin(value)[2:]
    if len(bin_val) > length:
    self.value = bin_val[-length:]
    else:
    self.value = '0'*(length - len(bin_val)) + bin_val
  else:
    if re.search("[0,1]{{{}}}".format(len(value)), value):
    self.value = '0'*(length - len(value)) + value
    else:
    raise ValueError("Wrong binary string!")

class extended_binary_mask:
  def __init__(self, string):
  if re.search("[0,1,*]{{{}}}".format(len(string)), string):
    self.value = string
  else:
    raise ValueError("Wrong extended binary mask!")

def operation_for_delta(binary_string_el, extended_binary_mask_el):
  if len(binary_string_el.value) == len(extended_binary_mask_el.value) and
    type(binary_string_el) == type(binary_string(1, 1)) and
    type(extended_binary_mask_el) == type(extended_binary_mask('1')):
  res = ''
  for i in range(len(binary_string_el.value)):
    if extended_binary_mask_el.value[i] == '*':
    res += '*'
    elif binary_string_el.value[i] == extended_binary_mask_el.value[i]:
    res += binary_string_el.value[i]
    else:
    res += binary_string_el.value[i]
  return extended_binary_mask(res)
  else:
  raise ValueError("Binary string length and extended binary mask length are not equal!")

def create_delta_set(extended_binary_mask_el):
```

```
  n = len(extended_binary_mask_el.value)
  V_n = list(range(2 ** n))
  result = []
  for el in V_n[1:]:
  if operation_for_delta(binary_string(el, n),
  extended_binary_mask_el).value == extended_binary_mask_el.value:
    result.append(el)
  return result
```

A.4 Програмна реалізація алгоритму пошуку високоімовірнісних усічених диференціалів

```
import pandas as pd
import numpy as np
import time
from heys import cipher
from collections import defaultdict
from itertools import product
from extended_binary_mask import binary_string

p = 0.00001
q = 0.0001

# read csv
start = time.time()
filename = 'kalyna_8.csv'
df = pd.read_csv(filename)
print(f'File {filename} was read.')
print(f'Time: {time.time() - start}')

# convert csv to array
start = time.time()
TDP = np.zeros([256, 256])
alpha_v = defaultdict(list)
def get_TDP(row):
 diff = row['Unnamed: 0'][1:-1].split(', ')
 diff = [int(i,2) for i in diff]
 TDP[diff[0]][diff[1]] = int(row['TDP(a, b)'])
 if int(row['TDP(a, b)']) != 0 and diff[1] != 255:
  alpha_v[diff[0]].append(diff[1])
 return row
df.apply(lambda row: get_TDP(row), axis=1)
del df
```

```
TDP = TDP / 256
print(f'csv was converted to array.')
print(f'Time: {time.time() - start}')

alpha_v[0] = [0]

def permutations_layer(diff):
 binary_string_diff = [binary_string(i) for i in diff]
 # print(binary_string_diff)
 result_binary_string = [['0' for i in range(8)] for j in range(8)]
 for i in range(len(binary_string_diff)):
  for j in range(len(binary_string_diff[i].value)):
   result_binary_string[j][i] = binary_string_diff[i].value[j]
 for i, e in enumerate(result_binary_string):
  result_binary_string[i] = ''.join(e)
 return [int(i, 2) for i in result_binary_string]

def round_sim(initial_alpha, prob_alpha):
 alpha = initial_alpha
 s_box_layer_res = list()
 for i in alpha:
  s_box_layer_res.append(alpha_v[i])
 s_box_layer_res = list(product(*s_box_layer_res))

 # print(s_box_layer_res)

 prob = list()
 for beta in s_box_layer_res:
  temp = prob_alpha
  for index, beta_i in enumerate(beta):
   temp *= TDP[alpha[index]][beta_i]
  if temp >= q:
   prob.append(temp)

 for_perm_layer = list()
 prob_res = list()
 for i, e in enumerate(prob):
  if e >= p:
   for_perm_layer.append(s_box_layer_res[i])
   prob_res.append(e)
 del s_box_layer_res
 del prob

 perm_layer_res = list()
 for i in for_perm_layer:
```

```
  perm_layer_res.append(permutations_layer(i))
 del for_perm_layer

 return perm_layer_res, prob_res


initial_alpha = [0, 0, 0, 0, 0, 1, 1, 1]
prob_alpha = 1
prob_alpha = 0.0546875

# prob_alpha = 0.010444641113281 25
# initial_alpha = [0, 0, 0, 0, 0, 128, 128, 128]

first_round_res, first_round_prob = round_sim(initial_alpha, prob_alpha)
print(first_round_res)
print(first_round_prob)
print(len(first_round_res))
print(len(first_round_prob))

# second_round_res = list()
# second_round_prob = list()
# for i, diff in enumerate(first_round_res):
#  temp_res = list()
#  temp_prob = list()
#  res, prob = round_sim(diff)
#  prob = list(np.array(prob) * first_round_prob[i])
#  for index, el in enumerate(prob):
#   if el >= p:
#    temp_res.append(res[index])
#    temp_prob.append(el)
#  second_round_res += temp_res
#  second_round_prob += temp_prob

# print(f'Second round: {len(second_round_prob)}')
```